Media Engineering and Technology Faculty
German University in Cairo

Social Listening for Social Good : Analysis of Public's Perception of Government's Policies

Bachelor Thesis

Author:	Omar Sherif Ali Hassan
Supervisors:	Prof. Mervat Mustafa Fahmy Abuelkheir

Submission Date: 1 June, 2023

Media Engineering and Technology Faculty
German University in Cairo

Social Listening for Social Good : Analysis of Public's Perception of Government's Policies

Bachelor Thesis

Author:	Omar Sherif Ali Hassan
Supervisors:	Prof. Mervat Mustafa Fahmy Abuelkheir

Submission Date: 1 June, 2023

This is to certify that:

(i) the thesis comprises only my original work toward the Bachelor's Degree

(ii) due acknowledgment has been made in the text to all other material used

Omar Sherif Ali Hassan
1 June, 2023

Acknowledgments

The completion of this thesis would not have been possible without the unwavering support and kindness of my friends and family. I offer them my heartfelt gratitude for their encouragement. Additionally, I would like to extend my sincerest thanks to my supervisor, Dr. Mervat Abu ElKheir, for her invaluable guidance and insights throughout the process. It has been an honor and a privilege to work under her mentorship, and I am truly grateful for the opportunity.

Abstract

In recent years, social media platforms have become increasingly popular for individuals to express their thoughts and opinions on various topics and situations. Monitoring sentiment and understanding the evolution of topics is crucial for governments to identify negative sentiments and respond promptly. In this study, we developed a sentiment analysis ensemble model architecture consisting of 4 different Transformers namely: MARBERT, CaMel-bert-DA, CaMel-bert-Mix, and Alanzi. This ensemble architecture was followed by a final classification layer, consisting of a three-layer feed-forward neural network. The output of each transformer applied on its logits a formula, and the resulting logits were then summed before applying the softmax activation function. Evaluating the model's performance on the sentiment analysis test dataset, an impressive test accuracy of 83%.

To analyze the temporal trends of sentiment, we applied the LSTM model using a sliding window to time-stamped tweets related to English News, which generated significant discussions on social media. That is then translated to Arabic by a translation model. we plotted the sentiment arc and observed our results with the original results.

We explored the effectiveness of BERTopic, a topic modeling technique, in comparison to LDA and NMF techniques. By employing various pre-trained Arabic language models as embeddings, we conducted topic modeling and aspect-based analysis. The results highlight that BERTopic and NMF achieved comparable and competitive outcomes, demonstrating their capability to capture meaningful topics effectively. However, LDA exhibited poor performance in generating coherent and informative topics. These findings emphasize the superiority of BERTopic and NMF over LDA in topic modeling tasks.

Contents

Chapter 1

Introduction

In today's digital age, citizens expect government services to be fast, efficient, and responsive to their needs. To meet these expectations, governments around the world are increasingly turning to social listening as a way to monitor and analyze citizen feedback in real-time. Social listening involves collecting and analyzing feedback from a variety of sources, including social media, official reviews, and citizen complaints, among others[33]. Governments worldwide are increasingly using social listening as a tool to monitor and analyze citizen feedback in real-time, with the aim of enhancing the quality of government services.

In Egypt, a similar approach has been taken by Egypt's Ministry of Communications and Information Technology(MCIT) to promote active community participation through a centralized online platform for public complaints, known as the shakwa.eg . This platform serves as a one-stop-shop for citizens to voice their concerns, suggestions, appreciation, and requests for information on a wide range of topics, including but not limited to infrastructure issues, education, and public services. By providing a convenient and accessible platform for citizens to voice their concerns and feedback, the (MCIT) aims to enhance citizen engagement and promote a more responsive and accountable government.In addition Egypt's (MCIT) want to utilize advanced analytical tools to analyze tweets in real-time, enabling policymakers to gain a more diverse and immediate understanding of public perceptions towards government policies and services. Through this approach, the (MCIT) aims to facilitate more informed decision-making and enhance responsiveness to citizen feedback. The MCIT's media center is leveraging data-driven insights to pave the way for a smarter, more citizen-centric approach to governance.To further enhance its analytical capabilities, Egypt's Ministry of Communications and Information Technology (MCIT) is leveraging sentiment analysis, forecasting, and topic modeling techniques in its analysis of tweets. By incorporating these advanced techniques, the MCIT is able to not only gain real-time insights into public perceptions towards government policies and services, but also forecast future trends and identify key topics and themes of interest to citizens. This approach enables the MCIT to develop more targeted and effective strategies for addressing citizen concerns and promoting greater engagement and accountability in government.

The combination of sentiment analysis and forecasting can further enhance the usefulness of Twitter analysis for government organizations. By forecasting future trends in public sentiment towards specific social issues, government organizations can anticipate potential areas of concern or opportunities for improvement in policy and communication strategies. This proactive approach enables organizations to take timely and effective actions to address emerging issues and promote positive public sentiment. The ability to forecast public sentiment can also help government organizations to prioritize their resources and efforts, ensuring that they are focused on the most pressing issues facing the public. Together, sentiment analysis and forecasting can provide valuable insights that enable government organizations to engage with the public in a more informed and effective manner[50]. In addition to sentiment analysis and forecasting, topic modeling can be a valuable technique for government organizations analyzing tweets[29]. Topic modeling allows organizations to identify and extract key topics and themes from large volumes of tweets, providing insights into the issues and concerns that are top of mind for the public. By understanding the most relevant topics and themes, government organizations can develop targeted policies and communication strategies that are tailored to the needs and priorities of the public.

Analyzing Arabic tweets presents a unique set of challenges that make sentiment analysis, forecasting, and topic modeling particularly relevant to Egypt's Ministry of Communications and Information Technology (MCIT). Firstly, the Arabic language is complex and nuanced, with multiple meanings and interpretations for words and phrases. This can make accurately gauging public sentiment a difficult task. Additionally, Arabic is a highly inflected language, with complex grammatical rules and syntax that can pose challenges for natural language processing algorithms. Finally, Arabic Twitter users frequently use dialects and informal language, which can make standard machine learning techniques less effective. By leveraging advanced analytical tools and techniques, the MCIT can overcome these challenges and gain deeper insights into public perceptions towards government policies and services, enabling more effective and responsive governance.

1.1 Motivation and Objectives

The significance of Sentiment Analysis has resulted in numerous research studies in this area. However, most of these studies have concentrated on English and other European languages, and only a few have tackled Sentiment Analysis in complex languages such as Arabic. Nonetheless, due to the surge in Arabic-speaking internet users and the remarkable expansion of Arabic digital content, Sentiment Analysis in this language has become a subject of great interest to many researchers in recent years.[23]

The lack of advanced techniques litowardske natural language processing (NLP) and state of art Transformers in the media center system in Egypt is a major challenge that could limit the full potential of social listening for social good.

As a result, My thesis will address the lack of advanced sentiment analysis techniques for the Egyptian dialect in the media center system targeting Twitter groups.

The primary goal of this thesis is to address the challenge of handling Arabic language and Egyptian dialect in social listening, this thesis aims to develop and implement advanced techniques for sentiment analysis, forecasting, and topic modeling. The goal is to create a highly accurate Ensemble sentiment analysis model that can assign sentiments to event-based time-series data, and track how these sentiments change over time. Furthermore, we will use forecasting techniques to predict future sentiments and identify potential areas of concern or dissatisfaction among the public, allowing for proactive measures to be taken. Finally, we will explore topic modeling to extract underlying themes and topics from social media data and identify emerging trends in public opinion.

1.2 Outline

This thesis consists of 5 chapters that focus on the application of sentiment analysis, forecasting, and topic modeling techniques in the analysis of Twitter data.

- Chapter 1, “Introduction” provides an overview of the objectives and scope of the research.
- Chapter 2, “Background” provides a brief introduction to the techniques used in sentiment analysis, forecasting, and topic modeling, as well as related work on sentiment tracking.
- Chapter 3, “Methodology” describes the proposed approach for conducting sentiment analysis, forecasting, and topic modeling using Twitter data.
- Chapter 4, “Experimentation Results” presents a description of the datasets used and the different experiments conducted to test the proposed approach. A discussion of the results and the limitations faced in this project are also presented.
- Chapter 5, “Conclusion Future work” summarizes the thesis and provides suggestions for future work, including the potential for expanding the use of sentiment analysis, forecasting, and topic modeling techniques to other social media platforms and domains.

Chapter 2

Background

Sentiment analysis have emerged as important areas of research in the field of natural language processing (NLP). Sentiment analysis aims to classify text based on the writer's attitude towards a particular topic, product, or service, whereas emotion analysis goes beyond sentiment and attempts to identify specific emotions expressed in the text, such as happiness, sadness, anger, or fear. These techniques have numerous applications in various domains, including marketing, social media monitoring, customer feedback analysis, and political science. [28]

In Sentiment and Emotion Analysis [23], an opinion is defined by making reference to a quintuple (o; a; h; t; so;) that consists of:

- Object 'o', which is the opinion target. It can be a product, service, topic, issue, person, organization, or event.
- Aspect 'a', which is the targeted attribute of the object 'o'.
- Opinion holder 'h', which is the person or organization that expresses an opinion.
- Time 't': the moment in which this opinion is expressed.
- Sentiment orientation 'so' that indicates whether an opinion is positive, negative, or neutral.

2.1 Arabic Text Challenges

Arabic is the fifth most widely spoken language in the world and the official language of 25 countries, making it an important language. Arabic words have a complex morphological structure, with prefixes, suffixes, and infixes, resulting in a large number of word forms. This study [18] presents challenges for tasks such as tokenization, stemming, and part-of-speech tagging.

- Lack of Standardization: Arabic is not fully standardized, and there are many variations of the language across regions and dialects. This makes it difficult to develop NLP tools that work well across different Arabic dialects.

- The sparsity of Data: There is a lack of high-quality, large-scale Arabic language datasets, which hinders the development of effective NLP models.

- Ambiguity: Arabic has a high degree of lexical and syntactic ambiguity, which makes it challenging to accurately interpret the meaning of the text. ex: ”أحببت بيته” can have two possible meanings: ”I loved his house” and ”I loved his poetry”

- Diacritization: Arabic script is written without vowels, and diacritization (adding vowel marks) is necessary for accurate text analysis. However, diacritization is often incomplete or inconsistent in Arabic text, which makes it difficult to process accurately. ex: The word آكتبْ in Arabic means ”books” or ”he wrote”. Without diacritization, it is impossible to tell which meaning is intended.

- Right-to-Left Writing Direction: Arabic is written from right to left, which requires special handling in NLP tasks such as tokenization and parsing.

- Limited Resources: There is a shortage of Arabic language processing resources, such as annotated corpora, tools, and software, which limits the development of Arabic NLP applications.

- Inconsistent name spelling (ex: Syria in Arabic can be written as “سورية” and “ سوريا”)

- Grammatical gender variation: all nouns, animate and inanimate objects are classified under two genders either masculine or feminine (ex: “كبيرة” and “كبير”

- Dual form “”, which can have multiple forms (ex“قلمان” - “qalamAn” or “قلمين” - “qalamyn” meaning “two pencils”)

Table 2.1: Sentiment Analysis Pipeline

Step	Description
Data Collection	Tweets are collected using the Twitter API or Data Sources
Preprocessing	Text are cleaned, stemming tokenized, and stop words are removed
Feature Selection	chooses an optimal subset of features
Feature Extraction	Word embeddings are created using GloVe or Word2Vec
Model Training	A Machine Learning model / Neural Network / Transformers is trained on the labeled data
Prediction	The trained model predicts the sentiment of new tweets
Evaluating the Model	Assess the trained model predictions based on Text Classification Criteria

2.2 Data Acquistion

There are numerous Arabic speakers across the world who use different varieties of Arabic based on their regions. However, Modern Standard Arabic (MSA) is the only standardized variety. Social media is widely used, and it is in this domain where local varieties of Arabic are commonly used, but resources for these varieties are limited. In March 2017, it was estimated that there were 11.1 million monthly active Twitter users in the Arab region, generating 27.4 million tweets per day according to Weedoo.1 Despite this high usage, Arabic, especially dialects, still lacks efficient resources for NLP tasks [39].

2.3 Text-Preprocessing

The following section describes text pre-processing and filtering, which are necessary when we talk about comments because ordinary people post comments directly without any review from social network administrators or owners. That is why comments and navigators' feedbacks are not Structured. Further, they may contain a mixture of words and characters and may also have spelling mistakes. In addition, since tweets are the focus of this work, we should pay attention to the special properties and symbols they have such as , @, URLs, usernames, etc. Almost all papers focusing on Twitter alter such orthography parts. With the removal of usernames, hyperlinks, URLs, repeated letters, etc., we can reduce the number of the features by about 54% [31][24]. The removal of the usernames had the largest impact on the reduction, which is plausible since the uniqueness of the usernames creates many new features.

- Letter normalization: unifying the letters that appear in different forms. We replace [إ,آ, أ,] with ا, [ة] with ه and [ي, ئ] with ى (Darwish et al., 2014)[25] .
- Suppression of diacritics
- Cleaning text: remove HTML line breaks and markup, remove unknown characters, diacritics, punctuation, and extra spaces.
- Replace all URLs with [رابط], emails with[بريد], mentions with [مستخدم]
- replace emojis with their semantic in-text "red heart" becomes حب
- Suppression of elongations: Overuse of characters and repetition of vowels, which might be used to stress or emphasize certain words or feelings. so we will decrease only the overuse to 2 characters instead of many characters of the word so that we don't lose the intensification كثييييير becomes كثيير.
- Unicode normalization is the process of transforming different representations of the same text into a single, standardized form. In Arabic text, there are four possible forms of normalization: NFC (Normalization Form Canonical Composition), NFD (Normalization Form Canonical Decomposition), NFKC (Normalization Form Compatibility Composition), and NFKD (Normalization Form Compatibility Decomposition). In NFC, "لآ" would be replaced with the precomposed character "" (U+FEFB), while in NFD, it would be represented as "ل" (U+0644) followed by "آ" (U+0622). NFC is the preferred form for Arabic text, as it is the most commonly used form and provides the most consistent and predictable behavior.[31]
- Keyboard mistyping, which can result in missing or extra spaces or confusion between characters that are located near each other on the keyboard.
- Phonetic errors based on the sounds of a language, such as homophones (two words that sound the same) or spoonerisms (switching two letters). It's important to consider these errors when analyzing the sentiment of Arabic tweets.

- Stop word filtering is important in many NLP tasks to reduce the problem's dimensionality and fall within the neutral polarity and hence do not add any value to the polarity decision.

2.4 Feature Extraction and Text Representation

Feature extraction and text representation are fundamental tasks in Natural Language Processing (NLP) that play a crucial role in transforming raw textual data into meaningful and machine-understandable representations. Feature extraction involves extracting relevant information or patterns from text, while text representation aims to encode the extracted features into numerical or vector representations that can be processed by machine learning algorithms. In this section, we will explore the concepts, and methods for feature extraction and text representation in NLP, shedding light on the key techniques used in these processes and their practical implications.

2.4.1 Bag-of-words

Count Vectorization

Count vectorization is a common technique used in natural language processing (NLP) for converting text data into a numerical representation that can be processed by machine learning algorithms. It involves transforming text documents into a matrix of token counts, where each row represents a document and each column represents a unique token (e.g., word or n-gram), with the corresponding cell in the matrix indicating the frequency of that token in the document. Count vectorization is a simple and effective way to represent text data for various NLP tasks, including sentiment analysis. Count vectorization has several benefits that make it helpful for sentiment analysis based on literature [48]. Firstly, it captures the frequency of occurrence of each token in a document, which can provide valuable information about the sentiment expressed in the text. For example, frequently occurring positive or negative words can be indicative of the sentiment conveyed in a document. Secondly, count vectorization is relatively simple and computationally efficient, making it suitable for processing large volumes of text data. It also preserves the order of words in the text, which can be important for sentiment analysis as the sequence of words can impact the sentiment expressed. Additionally, count vectorization allows for the inclusion of context-specific features, such as n-grams, which can capture more complex linguistic patterns and improve the performance of sentiment analysis models. Count vectorization also has some limitations. One limitation is that it does not capture the semantic meaning of words or the relationships between words. For example, it treats each occurrence of a word as a separate feature, without considering synonyms or related words. Another limitation is that it can be sensitive to the frequency of stop words or common words, which may not carry much sentiment information but can be heavily represented in the count matrix. Additionally, count vectorization may not be effective for capturing sentiment in short or noisy text data, such as social media posts or product reviews with limited text.

TF-IDF Vectorization

Another approach for representing documents is the vector space model, which is often used to calculate the similarity between documents. In this model, each document is represented as a vector of feature weights. These feature weights can be calculated using various methods, including the TF-IDF (Term Frequency-Inverse Document Frequency) weighting scheme.

The idea behind TF-IDF is that some words may occur frequently in many documents, but they may not necessarily be important, such as stopwords. The TF-IDF weighting scheme combines the individual frequency of each term or feature (TFf,d) in a document with the inverse frequency of the feature (IDFf) in the entire collection of documents [48].

$$TF - IDF(f, d, D) = TF_{f,d} \times IDF_f \tag{2.1}$$

where TF-IDF(f,d,D) represents the TF-IDF value for feature f in document d of the document collection D.TF(f,d) represents the term frequency of feature f in document d.IDF(f) represents the inverse document frequency of feature f in the entire collection of documents D.

2.4.2 Word Vectorization (Word2Vec)

Word embeddings are dense vector representations of words that capture the semantic and syntactic properties of the words.

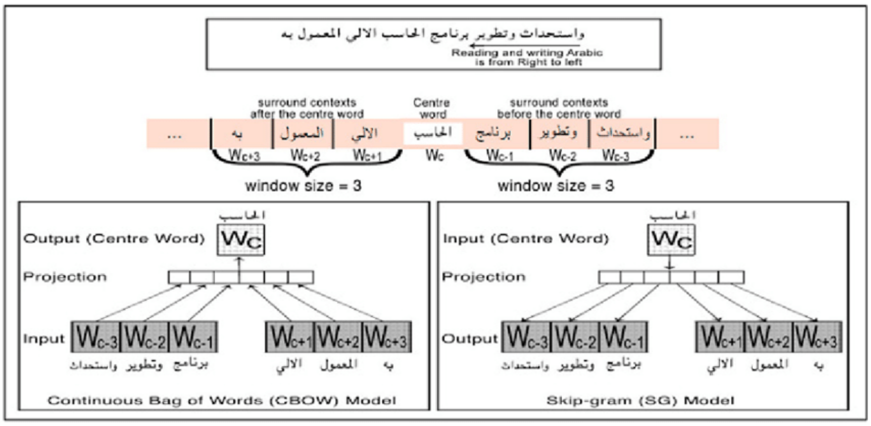


Figure 2.1: CBOW vs Skip Gram Architecture.Adapted from : [5]

Skip Gram Architecture

A technique that aims to predict the context words given the target word. The target word is used as input to a neural network that predicts the context words [42]. Rather than simply predicting the next word in the sentence, this model attempts to classify a word based on another word in the same sentence, using a log-linear classifier with a continuous projection layer. To achieve this, the model predicts words within a certain range before and after the current word. Increasing the range of the model can improve the quality of the resulting word vectors, but this also increases computational complexity. To mitigate this, the model assigns less weight to more distant words, as they are typically less related to the current word.

$$Q = C \times (D + D \times \log_2(V)) \tag{2.2}$$

Where Q represents the computational complexity, C is the number of classes, D is the embedding dimension, and V is the vocabulary size.

In this study [6], the Authors used Word Embeddings that was built using a corpus of 250M unique Arabic tweets; this makes it the largest Arabic word embeddings, The tweets were collected over different time periods between 2013 and 2016 to ensure the coverage of different topics. The large and diverse corpus ensures that many dialects are covered which would help in reducing the effect of dialectal variation and helped them achieve better results.

Continous Bag Of Words (CBOW) Architecture

A technique that aims to predict a target word based on the context words that surround it [42]. The context words are used as input to the neural network, and the target word is the output. The context words are represented as one-hot vectors and passed through a shared embedding layer, which maps the input vectors to a low-dimensional space where words with similar meanings are closer together.

$$Q = N \times D + D \times \log_2(V) \tag{2.3}$$

Where Q represents the computational complexity, C is the number of classes, D is the embedding dimension, and V is the vocabulary size.

According to the paper [5], Word2Vec was utilized to find similar words, and a large Arabic corpus containing 1.5 billion words was used to incorporate as many Arabic dialects and words as possible. The authors described the pre-processing steps for the corpus, constructed Word2Vec models, and selected the best one. The best model was then used to build an Automatic Arabic Lexicon that was used with different Machine Learning techniques. The lexicon was also used in Convolutional Neural Networks to expand the vocabularies, leading to an increase in sentiment classification accuracy for the Arabic Health Services dataset (AHS) from 0.85 to 0.92 for the Main dataset and from 0.87 to 0.95 for the Sub-dataset. Moreover, the paper reports an improved accuracy of 0.92 compared to the authors' previous results , which were 0.90 on the Main-Dataset.

2.5 Sentiment Analysis Models

Sentiment analysis models play a crucial role in Natural Language Processing (NLP) by enabling the automated identification and classification of sentiments expressed in textual data. These models provide insights into the emotional tone and subjective opinions of individuals towards specific topics or products. Discovering effective sentiment analysis models involves the exploration and development of innovative techniques that can accurately interpret and categorize sentiment, contributing to a wide range of applications such as social media monitoring, market research, and customer feedback analysis.

2.5.1 Levels of sentiment analysis

Sentiment Analysis can be investigated at three levels of granularity, namely document level, sentence level, and aspect level :

- Document level: the whole document expresses a subjective opinion on a single Entity that has an opinion holder ex: a blog post or an article.
- Sentence level: useful when a document contains multiple opinions on different entities or when the sentiment expressed towards a single entity changes throughout the document.
- Aspect level: provide more detailed insights into the specific factors that contribute to the overall sentiment expressed towards an entity. For example, in a product review, aspect-level analysis can identify the sentiment expressed toward specific product features such as design, functionality, or price. In their study [9](Al-Smadi et al., 2018), the authors developed an aspect-based sentiment analysis system that utilizes a Bi-LSTM and conditional random field (CRF). This system was applied to a dataset of hotel reviews written in Arabic, and was able to achieve an F-score of 70 percent, demonstrating the effectiveness of their approach in the Arabic language domain.

2.5.2 Supervised Learning Approaches

Supervised sentiment analysis involves training a machine learning model on a labeled dataset of texts that have been manually annotated with sentiment labels.

Machine Learning Text Classifier Techniques

Machine Learning with text is a subfield of Natural Language Processing (NLP) that focuses on developing algorithms and models that can analyze, understand and generate human language. Text is a very common type of data that can be found in various forms such as written documents, social media posts, customer reviews, and more. The main goal of Machine Learning with text is to create models that can automatically process and understand natural language, allowing computers to interact with humans more effectively and efficiently.

Multinomial Naive Bayes(MNB) is a probabilistic algorithm that is often used in text classification tasks, including sentiment analysis. The algorithm is based on Bayes' theorem, which states that the probability of a hypothesis (in this case, a class label) given some observed evidence (in this case, a text document) is proportional to the probability of the evidence given the hypothesis, multiplied by the prior probability of the hypothesis. In recent studies [2][20] have proposed various extensions and modifications to the basic MNB algorithm to address some of these limitations. For example, some researchers have proposed using n-grams instead of individual words to capture some of the context and order information in a document. Others have proposed using more sophisticated feature selection and weighting techniques, such as TF-IDF , to improve the relevance and discrimination power of the features. Some have also proposed using more complex Bayesian models, such as hierarchical or Bayesian networks, to capture dependencies between features and classes.

Support Vector Machines (SVMs) The basic idea behind SVMs is to find the optimal hyperplane that separates the data into different classes. In other words, given a set of training data points, SVMs aim to find the best linear boundary that maximizes the margin between the two classes. The margin is defined as the distance between the hyperplane and the closest points from each class. The SVM algorithm finds the hyperplane that maximizes this margin, making it less likely to misclassify future data points.

In cases where the data is not linearly separable, SVMs make use of kernel functions to transform the input data into a higher-dimensional space where the data can be separated more easily. The choice of kernel function depends on the problem at hand, but some popular ones include linear, polynomial, radial basis function (RBF), and sigmoid. According to the study conducted by SVM experts [10], the results indicate that SVM-RBF algorithm achieved the highest f-measure of 88.8% when applied to an unbalanced dataset with two classes.

Decision Tree The process of generating a decision tree is usually accomplished using a recursive partitioning algorithm. At every non-terminal node of the tree, this algorithm divides the data based on the attribute that provides the greatest discrimination between the filtered cases.

In this study [32], Facebook was utilized to gather training data for conducting sentiment analysis. Three separate corpora were created for supportive, attacking, and neutral comments regarding various posts. Various feature sets were experimented with, and improvement was made by incorporating similarity and sentiment words features. Multiple classifiers were used, including support vector machines, naive bayes, and decision trees. The support vector machine classifier achieved the highest accuracy among the models, reaching up to 73.4% accuracy on the test set.

ML Classifier	**Strengths**	**Weaknesses**
Multinomial Naïve Bayes (MNB)	• is a simple and efficient algorithm that is easy to implement and can be trained quickly on large datasets. • It performs well in many text classification tasks, including sentiment analysis, spam filtering, and topic classification. • It is less prone to overfitting than some other machine learning algorithms, due to its smoothing techniques. • It works well even with high-dimensional data, as it can handle large feature spaces with many variables.	• The independence assumption of the algorithm may not hold in some cases, especially when dealing with long documents or complex text structures. • It does not consider the order or context of words, which may limit its performance in tasks that require a more nuanced understanding of the text. • It may not perform well with rare or unseen features, as it estimates probabilities based on the frequency of features in the training data. • It assumes that all features are equally important, which may not hold in some cases, especially when dealing with noisy or irrelevant features.

Support Vector Machines (SVM)	• Robust to outliers: SVM is less sensitive to outliers in the training data compared to other machine learning algorithms. It uses a margin-based approach to find the optimal decision boundary, which reduces the impact of outliers on the model's performance. • Ability to handle high-dimensional data: SVM can handle data with a large number of features or dimensions, making it suitable for problems with complex datasets. It can effectively capture patterns in high-dimensional data, such as image and text data, making it widely used in image recognition, text classification, and natural language processing tasks. • Versatility in kernel functions: SVM allows for the use of different kernel functions, such as linear, polynomial, and radial basis function (RBF) kernels, which can be tailored to specific datasets and problem domains. This flexibility allows SVM to capture nonlinear patterns in data, making it a powerful tool for solving complex problems.	• Memory-intensive: SVM requires storing all support vectors, which are the data points that define the decision boundary, in memory during training. This can be memory-intensive and may pose challenges when working with large datasets. • Computational Complexity: SVMs can be computationally expensive, especially when dealing with large datasets, as the training time increases exponentially with the number of samples. This can be a limitation in real-time or high-speed applications where fast prediction times are required. • Binary Nature: SVMs are inherently binary classifiers, and their extension to multi-class classification involves either training multiple binary classifiers using one-vs-one or one-vs-rest strategies or modifying the SVM formulation to handle multiple classes directly. These approaches can result in increased complexity and reduced interpretability compared to other multiclass classification techniques.

K-Nearest Neighbour (KNN)	• Non-parametric: KNN does not make any assumptions about the underlying distribution of data, making it suitable for handling data with complex patterns or distributions. • Interpretability: KNN provides interpretable results, as the predicted sentiment class for a given text is based on the majority class of its k-nearest neighbors, making it easy to understand and explain the predictions.	• Computational cost: KNN can be computationally expensive, especially with large datasets, as it requires calculating the distances between the query instance and all the training instances in the dataset, which can be time-consuming and resource-intensive. • Curse of dimensionality: KNN performance can degrade when dealing with high-dimensional data, as the distance between instances tends to converge, and the notion of ”nearest neighbors” becomes less meaningful. This can result in reduced accuracy and increased computational cost.
Logistic Regression	• Simplicity: Logistic Regression is a relatively simple algorithm that is easy to implement and understand. It does not require complex calculations and can be applied to large datasets without much computational overhead. • Interpretability: Logistic Regression provides interpretable results, allowing for easy analysis and understanding of the relationship between the input features and the predicted sentiment classes. It can provide probability estimates for each class, which can help in decision making.	• Linear Decision Boundary: Logistic Regression assumes a linear relationship between the input features and the predicted sentiment classes. This can limit its ability to capture complex nonlinear patterns in the data, resulting in lower accuracy for sentiment analysis tasks with complex data distributions. • Limited Modeling Flexibility: Logistic Regression may not be able to capture complex sentiment patterns that require more sophisticated modeling techniques, such as deep learning algorithms.

Decision Tree	• Feature Importance: Decision trees can provide feature importance measures, which can help identify the most relevant features for sentiment analysis. This can provide insights into which words or phrases are most indicative of different sentiment categories, aiding in feature selection and model interpretation. • Non-linearity: Decision trees are capable of capturing non-linear relationships between features, which can be useful for sentiment analysis tasks where sentiment expression can be complex and non-linear.	• Lack of global optimization: Decision trees are greedy algorithms that make local decisions at each split, which may not always result in the optimal global solution. This can lead to suboptimal decision paths and reduced overall accuracy. • Sensitivity to data changes: Decision trees can be sensitive to small changes in the input data, which may result in different tree structures and predictions. This can make decision trees less stable and reliable compared to other machine learning algorithms.

Random Forest	• High accuracy: Random Forest is known for its high accuracy in predicting multiclass sentiment analysis tasks. It can effectively handle data with multiple classes and provide accurate predictions for sentiment labels. • Robust to overfitting: Random Forest is less prone to overfitting compared to other machine learning algorithms. It builds multiple decision trees and combines their predictions, which helps to reduce overfitting and improve the model's generalization ability.	• Lack of interpretability: Random Forest is an ensemble method that combines multiple decision trees, which can make it difficult to interpret the model's predictions and understand the underlying feature importance for sentiment analysis. This may limit the model's explainability in certain applications. • Computationally expensive: Random Forest builds multiple decision trees, which can be computationally expensive, especially when dealing with large datasets. Training and predicting with Random Forest may require more time compared to other algorithms, which may impact real-time applications.

XGBOOST	• Feature importance estimation: XGBOOST provides a built-in feature importance estimation mechanism, which allows users to understand the contribution of each feature in the prediction process. This can be useful in interpreting the model's predictions and identifying key features that are driving sentiment predictions. • Robustness to overfitting: XGBOOST incorporates regularization techniques such as L1 and L2 regularization, which helps prevent overfitting, a common issue in sentiment analysis where the model may memorize the training data rather than generalize from it.	• Sensitivity to hyperparameters: XGBOOST requires careful tuning of hyperparameters such as learning rate, tree depth, and regularization parameters to achieve optimal performance. Selecting the right hyperparameters can be challenging and may require experimentation and expertise. • Lack of interpretability: Although XGBOOST provides feature importance estimation, the model itself is a complex ensemble of decision trees, making it less interpretable compared to simpler models like logistic regression. Understanding the internal workings of XGBOOST and explaining its predictions to non-experts may be difficult.

Deep Learning Text Classifier Techniques

is a subfield of machine learning that involves the use of artificial neural networks with multiple layers. These neural networks are designed to learn from data in a hierarchical manner, with each layer learning increasingly complex features or representations of the input data. The term ”deep” refers to the large number of layers in these networks, which can range from dozens to hundreds or even thousands.

Artificial Neural Network (ANN) undergoes a series of mathematical transformations and computations in the hidden layers of the network, ultimately producing an output of the value that represents the predicted sentiment label.

Feedforward Neural Network (DNN) are a type of artificial neural network in which the information flows in one direction, from the input layer to the output layer.

Convolutional Neural Networks Architectures (CNN) involves 1D convolutional layers to extract features from the text specifically from its word embeddings, followed by pooling(downsampling) and fully connected(Dense) layers for classification. New data points are mapped to a label based on their similarity(patterns) to previously labeled data. As depicted in this Figure 2.2

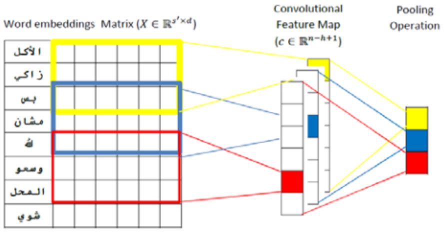

Word embeddings Matrix ($X \in \mathbb{R}^{s' \times d}$)
Convolutional Feature Map ($c \in \mathbb{R}^{n-h+1}$)
Pooling Operation
ا لأكل
ز اكي
بس
مشان
لله
وسعو
المحل
شوي

Figure 2.2: A typical CNN Layer Architecture, the Colored Boxes Represent the Size of a Filter Spatially Adapted from: [12].

RNN Figure 2.3 illustrates the contrast between Recurrent Neural Network (RNN) and Feed-Forward Neural Network (FNN). It shows that in FNN, data moves in only one direction, while in RNN, there is a loop. As depicted in this Figure 2.3.

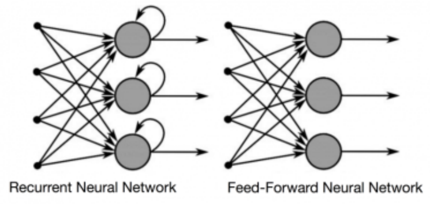
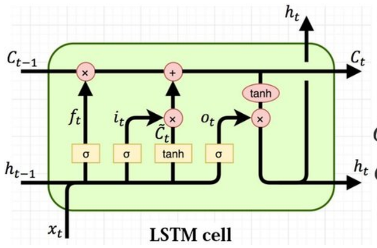


Figure 2.3: RNN vs FNN Adapted from: [27]

Long Short-Term Memory (LSTM) is a type of RNN that address the vanishing gradient problem by introducing memory cells and gates that control the flow of information within the network.



$$i_t = \sigma(x_t U^i + h_{t-1} W^i)$$
$$f_t = \sigma(x_t U^f + h_{t-1} W^f)$$
$$o_t = \sigma(x_t U^o + h_{t-1} W^o)$$
$$\tilde{C}_t = \tanh(x_t U^g + h_{t-1} W^g)$$
$$C_t = \sigma(f_t * C_{t-1} + i_t * \tilde{C}_t)$$
$$h_t = \tanh(C_t) * o_t$$

Figure 2.4: LSTM Structure and Equations Adapted from: [12].

The memory cell is a vector that stores information over time and can be updated or reset based on the input and the previous state of the network. The gates, which are composed of sigmoid and element-wise multiplication operations, control the flow of information into and out of the memory cell, as well as the forget gate, which decides which information to keep or discard.

According to this study [12] on Arabic sentiment classification, a new deep learning model was proposed that utilized a linear SVM classifier on top of a combination of CNN and Bi-LSTM. Unlike traditional models, the fully connected layer was replaced with an SVM classifier, which received embedded vectors extracted by CNN and Bi-LSTM. Results from experiments showed that this model was highly effective and outperformed other baseline and state-of-the-art models. Additionally, the proposed model showed a significant improvement compared to one of the state-of-the-art deep learning models. The results of the experiments showed that the model performed well, achieving an f1-score between 86 percent to 8 percent.

In this study [11] conducted binary sentiment analysis on Arabic texts. They first cleaned the input texts through preprocessing, followed by using a word embedding layer to convert the texts into numerical vectors to be input into the LSTM layer. Lastly, a SoftMax layer was used to predict the polarity of the text. The results of the experiments showed that the model performed well, achieving an accuracy between 80 percent to 82 percent.

In this study [45], the authors examined the performance of LSTM and Bi-LSTM models on Saudi dialectical tweets after applying some basic normalization techniques. They represented words as vectors using the CBOW model and reported that the Bi-LSTM model outperformed both the LSTM and SVM models, achieving an accuracy of 94%. Another study [8] investigated various deep learning models, including a combination of LSTM and CNN, to predict sentiment in tweets. They also explored the use of pre-trained word vectors from the CBOW and Skip-gram models introduced in [15]. The experiments showed that a two-layer LSTM model achieved the best results. Barhoumi et al. [1]utilized Bi-LSTM and CNN to evaluate different types of Arabic-specific embeddings and recommended using Arabic NLP tools to address the effect of the agglutinate and morphological richness of Arabic on word embedding quality. However, these NLP tools, such as lemmatization, light stemming, and stemming, have mostly been developed for MSA texts.

In Conclusion In this study [51], the authors compare deep learning models (such as LSTM and CNN) to other machine learning models in the context of binary and multi-class classification, using various datasets. The study concludes that deep learning models are more effective for larger datasets, while basic machine learning algorithms are more suitable for smaller datasets.

DL Classifier	**Strengths**	**Weakness**
Fead-Forward Neural Networks (FNN)	• Nonlinear mapping: Feedforward networks can model complex nonlinear relationships between the input text and its corresponding sentiment. This allows them to capture subtle nuances in language that might be missed by simpler models.	• Limited context awareness: Feedforward networks do not have a built-in understanding of context or long-term dependencies in text, which can limit their ability to capture the full meaning of a sentence or document. • Lack of interpretability: FFNNs are often considered ”black box” models, meaning that it can be difficult to interpret how the network makes its predictions. This can make it challenging to understand why the model has made a particular prediction, which can be a problem for applications where transparency and accountability are important.

Convolutional Neural Networks (CNN) Architectures	• CNNs are particularly effective at capturing local patterns in sequential data, such as the presence of certain words or phrases that contribute to sentiment. This makes them well-suited to sentiment analysis, where identifying key words or phrases that express positive or negative sentiment is often critical. • CNNs can learn complex features of the input data through hierarchical layers of convolutions and pooling operations. This means that they are able to extract useful features from raw text input, such as the presence of certain parts of speech or syntactic structures, without requiring manual feature engineering.	• CNNs may struggle to capture long-term dependencies in sequential data since they typically operate on fixed-length input sequences. This could be a limitation in sentiment analysis, where understanding the overall sentiment of a piece of text may require a broader context than what can be captured in a local window of words. • CNNs can be sensitive to the choice of hyperparameters such as the number of filters and filter sizes, which can impact their performance. Tuning these hyperparameters can be a time-consuming and resource-intensive process.

Recurrent Neural Networks (RNN)	• Ability to capture sequential information: RNNs are designed to handle sequential data and can capture long-term dependencies between words in a sentence. This makes them ideal for sentiment analysis, where the sentiment of a sentence can depend on the context and order of the words. • Memory: RNNs are able to store information from previous inputs using hidden states, which can help to capture the sentiment of a sentence by taking into account the sentiment of the previous words.	• Vanishing Gradient Problem: RNNs can suffer from the vanishing gradient problem, where the gradient becomes too small to propagate back through the network during training. This can cause the network to have difficulty capturing long-term dependencies, which can be problematic for sentiment analysis. • Computationally Intensive: RNNs can be computationally intensive, particularly when dealing with large datasets or complex models. This can make training and inference times slow and expensive.

2.5.3 Unsupervised Learning

is a type of approach that does not require labeled data to train a model for sentiment analysis. Instead, it relies on identifying patterns and structures in the data to determine sentiment.

Lexicon-Based Approach

The lexicon-based approach used in sentiment analysis can be classified into two main types: dictionary-based and corpus-based.

The dictionary-based approach involves searching for sentiment seed words with known positive or negative orientation, and then using a dictionary like WordNet to locate their synonyms and antonyms [30].

The corpus-based approach , uses a predefined seed list of sentiment words to find other words with domain-specific orientations, using statistical or semantic methods in a large corpus. This approach is used to adapt a general sentiment lexicon to a new domain-specific lexicon [14]. Bonta et al.[52] conducted a study comparing different lexicon-based libraries and tools, including SentiWordNet, TextBlob, and VADER. They found that VADER, which is specifically attuned to social media texts such as tweets[35], performed the best with an accuracy of 77.0% and an F1-score of 81.60%, based on a labeled set of 11,861 movie reviews from a movie review-aggregation website. Although sentiment lexicons provide knowledge on the prior polarity of a word, they suffer from poor recognition of sentiments due to the context-dependence of sentiment orientation or polarity in human language. For example, the word ”جيد” usually indicates a positive sentiment as in ”كان وقت جيد”, but it could imply a neutral sentiment as in ”عايز اجيب جيد”.[28]

2.5.4 self-supervised Transformers

Since the introduction of attention mechanism in machine learning, sentiment analysis techniques have evolved from recurrent neural networks to transformer models. Transformer-based models are encoder-decoder systems with attention. Attention mechanism has permitted models to consider only relevant parts of a given sequence. Making use of this feature in encoder-decoder architecture has impacted the performance of transformer models in several natural language processing tasks, including sentiment analysis.[36][19]

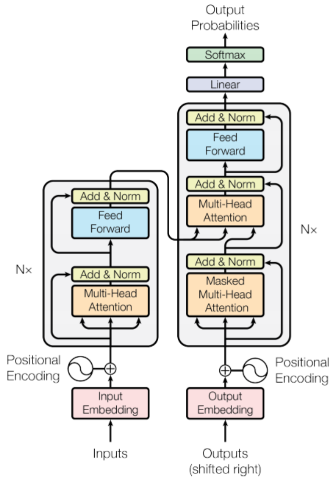


Figure 2.5: Transformer Architecture Adapted from: [19].

Types of Transformer

Transformer-based models are classified into three types, namely encoder-only, decoder-only, and encoder-decoder models :

Encoder-only models rely on only the encoder part, where a vector of the input sequence is fed into the first encoder block. The block consists of a bidirectional self-attention layer and a feed-forward layer, and the output is passed to the following encoder block, which consists of two layers that aim to enrich the embedding vector with contextual information. The final encoder block produces the last contextual encoding, making this type of transformer suitable for text classification and named entity recognition tasks. Some popular examples of encoder-only models are BERT, ELECTRA, and RoBERTA [19].

Decoder-only models rely solely on the decoder stack, which is composed of N identical decoder blocks. Each decoder block consists of three layers, with the first layer being a masked multi-head attention layer. The next layers include multi-head self-attention layers and a fully connected feed-forward network. Decoder-only models, also known as autoregressive models, are ideal for text-generation tasks. GPT (Generative Pre-trained Transformer) is one of the most widely used models trained with decoder-only architectures[19][46].

Encoder-Decoder models also known as sequence-to-sequence models, employ both the encoder and decoder blocks. The encoder block considers the entire sequence, while in the decoder block, only the words preceding a given word are considered. This type of transformer is best suited for tasks involving the input of a sequence of items, such as machine translation or question answering. Many encoder-decoder based models have recently been introduced, and they have shown promising results in various natural language processing tasks[19].

Transformer Language Models for Arabic In recent times, many transformer-based models have been developed and pre-trained to carry out Arabic sentiment analysis tasks. These models are primarily built on Bidirectional Encoder Representations from Transformers (BERT) framework, such as AraBERT, CAMeLBERT, Arabic ALBERT, and GigaBERT.[36]

In Conclusion The sentiment analysis of Modern Standard Arabic (MSA) and various Arabic dialects including Levantine, Egyptian, and Gulf was explored in these studies [37][36]. The study employed three-way classification based on positive, neutral, and negative scales using different algorithms like Naive Bayes classifiers (NB), Support Vector Machine (SVM), Random Forest Classifiers, and BERT model (Bidirectional Encoder Representations from Transformers). The BERT model produced the most favorable outcomes of accuracy score.

2.6 Ensemble Models

is a strategy that involves merging several machine learning models to enhance the learning model's efficiency and attain a better accuracy score than what can be obtained by a single model in the ensemble.

Bagging (Bootstrap Aggregation) multiple copies of a single model are trained on different subsets of the training data using a technique called bootstrap sampling. The predictions of each model are then combined through a majority vote to obtain the final prediction. Random forests is an extension of bagging that builds a large number of decision trees on randomly selected subsets of the training data. The final prediction is obtained by averaging the predictions of all trees.

Majority voting the final output of the ensemble model is determined by counting for each class the number of votes of multiple models. The class with the majority of votes is predicted.

Proposed method SUM In this study [36], the output is obtained by calculating the weighted sum of the probabilities from the same class, then for each class, the argmax operation is applied to find the class with higher probability value , and it achieved a high Accuracy score of 96.16 percent for Binary SA by summing CAMeLBERT and AraBERT prediction while considering considered 70 percent of the probability generated by CAMeLBERT model and 30 percent of the probability generated by AraBERT model and he used Majority Voting and achieved 95.65 percent .

2.7 Loss Functions

Loss functions play a fundamental role in natural language processing (NLP) tasks by quantifying the discrepancy between predicted outputs and ground truth labels. In NLP, where tasks such as text classification, machine translation, and sentiment analysis are prevalent, choosing an appropriate loss function is crucial for training and evaluating models effectively. The selection of a loss function depends on the specific NLP task and the desired model behavior. By understanding and utilizing suitable loss functions, researchers and practitioners can enhance the performance and accuracy of NLP models, ultimately enabling them to tackle complex language understanding and generation tasks more effectively.

2.7.1 Sparse Categorical Cross Entropy (SCCE)

loss function commonly used in training neural networks for natural language processing tasks such as multiclass classification [53].

In particular, Sparse Categorical Cross Entropy is used when the target variable is represented as integers (e.g., when using one-hot encoding) rather than as a binary or continuous value. It measures the difference between the predicted probability distribution and the true distribution of the target variable, with the goal of minimizing this difference during training.

The "sparse" part of Sparse Categorical Cross Entropy refers to the fact that the true labels are represented as integers rather than one-hot encoded vectors. This is useful in cases where the number of classes is very large, as it reduces the memory requirements for storing the true labels .

$$Loss = -\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{C} y_{ij}\log(\hat{y}_{ij}) \tag{2.4}$$

where N is the number of samples, C is the number of classes, y_{ij} is the true label for the i-th sample and j-th class (either 0 or 1), and $\hat{y}_{ij}$ is the predicted probability for the i-th sample and j-th class.

This formula computes the cross-entropy loss between the true distribution (encoded as integers) and the predicted distribution (encoded as probabilities). The loss is higher when the predicted distribution is far from the true distribution.

Intuitively, the formula penalizes the model more for predicting low probabilities for the true class and high probabilities for the wrong classes. By minimizing the SCCE loss during training, the model learns to make better predictions and improve its accuracy on the classification task.

2.7.2 Focal Loss

Focal Loss is to modify the standard cross-entropy loss function in such a way that it gives more emphasis to the difficult-to-classify examples. In a standard cross-entropy loss function, all examples are given equal weight, which may not be ideal for multiclass classification tasks, especially when the classes are imbalanced. Focal Loss addresses this issue by assigning higher weights to the misclassified examples, particularly those that belong to the minority classes.

$$L_{FL}(p_t) = -\alpha_t(1-p_t)^{\gamma}\log(p_t) \tag{2.5}$$

where p_t is the predicted probability of the correct class, α_t is the class balancing factor, and γ is the focusing parameter.

According to the [43] study, it is demonstrated that the focal loss approach results in the implicit maximization of entropy while minimizing the KL divergence between the target and predicted distributions. The study further highlights that this approach's design inherently regularizes network weights during the training process, which reduces the overfitting of negative log-likelihood (NLL) and ultimately enhances calibration.

2.7.3 Hyper-parameters Tuning

Hyperparameters are the settings of a machine learning algorithm that are not learned from the data but are set by the user. Examples of hyperparameters include learning rate, regularization strength, number of hidden layers, and number of neurons in each hidden layer.

Grid Search

GridSearchCV is a hyperparameter tuning method used in machine learning to systematically search for the optimal hyperparameters of a model.

GridSearchCV works by specifying a set of hyperparameters and their possible values and then training and evaluating the model for all possible combinations of these hyperparameters. It then selects the combination that gives the best performance on a validation set, which is a portion of the training data that is used to evaluate the model's performance.

The "grid" in GridSearchCV refers to the fact that the method searches over all possible combinations of hyperparameters in a grid-like fashion. For example, if we have three hyperparameters with two possible values each, GridSearchCV would train and evaluate the model for all 2 x 2 x 2 = 8 possible combinations of hyperparameters.

GridSearchCV is an important tool for ML researchers because it allows us to optimize the performance of our models without relying on intuition or trial and error. By systematically searching over a range of hyperparameters, we can find the optimal settings for our models and achieve the best possible performance.

Random Search CV

RandomSearchCV is a technique that allows me to search over a range of hyperparameters for my model by randomly selecting a combination of hyperparameters from a predefined range. This helps me to avoid the time-consuming process of trying out all possible combinations of hyperparameters.

To implement RandomSearchCV, First define a range of hyperparameters that I want to explore. This could include things like learning rate, batch size, number of epochs, and regularization strength. Then, use RandomizedSearchCV class, which takes in my machine learning model, the hyperparameter range, and the number of iterations I want to perform. The RandomizedSearchCV class will then randomly select a combination of hyperparameters from the range I defined and train the model using those hyperparameters. This process is repeated a specified number of times, and the best-performing set of hyperparameters is returned.

RandomSearchCV is a powerful technique because it can help me to find the best hyperparameters for my model in a relatively short amount of time. It is also flexible, as I can define any range of hyperparameters that I want to explore. This technique can help me to optimize my machine learning models and produce more accurate results, which is critical for the success of my thesis research.

2.8 Sentiment Forecasting

Time series data are a widely used data type in various domains and applications, including finance, demand and supply prediction, and health monitoring. One common task in time series analysis is to predict future values of the time series based on a historical vector of time series values T = [t1, t2, . . . , tN] RN.

Time series forecasting techniques can be classified into two broad categories: conventional statistical techniques and methods based on machine learning models. Although recurrent neural networks (RNNs) belong to the latter group, they are commonly used for time series forecasting[29].

Sentiment forecasting is an application of time series forecasting that involves predicting the sentiment of Arabic tweets, which can be positive, negative, or neutral. To accomplish this task, several machine learning and deep learning models have been developed. In this paper, we will discuss some of the popular models used for sentiment forecasting in Arabic tweets.

Regression models are a class of machine learning models that are used to predict a continuous target variable, given one or more input variables. In the context of sentiment forecasting, a regression model can be trained to predict the sentiment score of a tweet, which can range from -1 (negative) to 1 (positive), with 0 indicating a neutral sentiment. The input variables can be various features extracted from the tweet, such as the presence of certain words, hashtags, or emoticons. Some popular regression models used for sentiment forecasting in Arabic tweets include linear regression, logistic regression, and support vector regression.

On the other hand, deep learning models have been found to be highly effective in sentiment forecasting tasks due to their ability to learn complex representations of the input data. Some popular deep-learning models used for sentiment forecasting in Arabic tweets include Convolutional Neural Networks (CNNs), and Recurrent Neural Networks (RNNs).

This study [34] aims to address issues with using recurrent neural networks (RNNs) for forecasting, particularly for practitioners with limited knowledge of these techniques. Through systematic experiments, the study examines data preprocessing, hyperparameter configurations, RNN architectures, recurrent units, and optimizers. The study finds that the Stacked architecture combined with LSTM cells and the COCOB optimizer, fed with deseasonalized data in a moving window format, can be a competitive model across many datasets. The study also concludes that leveraging cross-series information can improve forecasting accuracy, and that RNNs are good candidates for forecasting, often outperforming statistical benchmarks. While fitting RNNs is not yet as straightforward as for some statistical benchmarks, the study and its released code framework provide important steps towards automating the process.

2.9 Topic Modeling

Topic modeling is a valuable tool for managing and comprehending large sets of text data. It allows for the identification of underlying topics that differ across documents within a corpus, aiding in summarizing and understanding the information. Two commonly used techniques for topic modeling are Latent Dirichlet allocation (LDA) and Non-Negative Matrix Factorization (NMF). While LDA employs a probabilistic approach, NMF utilizes a matrix factorization approach. Recently, new advanced techniques have emerged that employ BERT for topic modeling [7].

While Arabic topic modeling research has been developing over the years, its practical applications remain limited. In this section, we highlight recent works related to the topic modeling of the Arabic language. In this study [47] conducted a survey on various methods of topic detection and modeling techniques related to the Arabic domain. Their findings showed that most previous works utilized LDA for topic modeling, while recent works started to combine LDA with other techniques such as K-means clustering and word2vec embeddings. In a recent study,[44] used paragraph vectors to create fixed-length vector representations for each verse/sentence in the Quran. They evaluated the related clusters of verses against a tagged corpus to verify the relationships between Quranic verses, identify their associations, and address the concepts covered in each cluster. Additionally, in the realm of social media on the Arabic web, topic modeling research has been focusing on social media content. In this study [21] utilized NMF to discover the primary issues and topics discussed in hate tweets during the COVID-19 pandemic. NMF assisted them in identifying seven commonly discussed topics in hate tweets during the pandemic.

2.10 Summary

In recent years, sentiment analysis and forecasting have gained increasing attention in the field of natural language processing (NLP), especially for social media data. Arabic, as one of the most widely spoken languages in the world, has also been a focus of research in this area. In this literature review, we surveyed the state-of-the-art techniques for sentiment analysis and forecasting, as well as topic modeling, in Arabic sentiment.

We first discussed the various approaches used for sentiment analysis, including lexicon-based, machine learning-based, and hybrid methods. We highlighted the challenges of performing sentiment analysis in Arabic, such as the complexity of the language, the lack of annotated data, and the presence of dialectal variations. We then reviewed recent works that have attempted to overcome these challenges, including the use of transfer learning and neural networks.

Next, we explored sentiment forecasting, which aims to predict the sentiment of future events or trends. We discussed the different types of forecasting models, such as regression-based, and deep learning-based models.

Finally, we investigated the topic modeling techniques used for Arabic sentiment analysis, which aim to discover latent topics or themes in a given text corpus. We surveyed the most commonly used topic modeling algorithms, such as Latent Dirichlet Allocation (LDA) and Non-negative Matrix Factorization (NMF), and the latest algorithm like Bertopic and their applications in Arabic sentiment analysis.

Overall, this literature review provides a comprehensive overview of the current state-of-the-art techniques for sentiment analysis and forecasting, as well as topic modeling, in Arabic sentiment.

Chapter 3

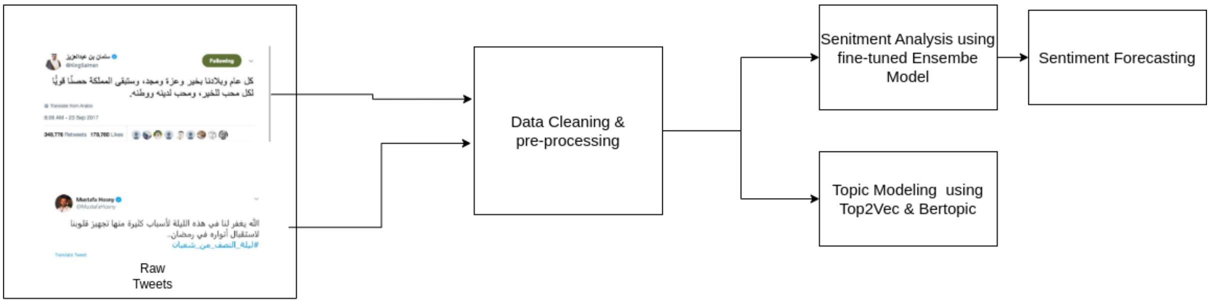
Methodology

Twitter is a social media platform that stands out as a valuable resource for research due to its vast user base and high volume of activity. With over 330 million active users[1] generating approximately 500 million tweets daily, Twitter provides a wealth of data encompassing a wide range of topics spanning different domains, such as commerce, health, and disaster management. The significance of Twitter in the current social landscape is steadily increasing, making it a valuable source for conducting experiments. Additionally, Twitter's data provides insights into universal human patterns, including emotional and social behavior, which makes it an ideal platform for studying a variety of phenomena.

The proposed method involves several steps. Firstly, a series of tweets related to the chosen topic is collected from Twitter. Next, the data is pre-processed to remove noise and redundant information, ensuring more accurate assessments. Then, a classification model is constructed by fine-tuning four different Arabic BERT models on a combined dataset of labeled tweets related to news, sarcasm, and emotions to predict sentiments. Each model is independently tested separately on the test set to determine its accuracy on unseen data. The accuracies are normalized so that they sum up to 1, and each accuracy is multiplied by its model's prediction Tanh(logits) to scale down the values into the -1 to 1 range because logits can be negative values). The models' predictions of each model are then summed together and passed through a softmax function to obtain our label, which is how our ensemble model works. we plan to conduct a comparative analysis of baseline machine learning classifiers and deep learning models, along with my ensemble model, to determine the most effective approach for the task at hand. By evaluating the performance of each model based on the evaluation methods that are discussed in this Section 4.9.

The sentiment scores are plotted against time to produce a sentiment arc. Additionally, we apply the Bertopic topic modeling algorithms to extract other topics discussed in the data and compare its results with 2 baseline models namely LDA and NMF. The architecture of this method for tracking sentiments is illustrated in Figure 3.1.

[1] Twitter Statistics: https://www.omnicoreagency.com/twitter-statistics/



سلمان بن عبدالعزيز
@KingSalman
Following

كل عام وبلادنا بخير وعزة ومجد، وستبقى المملكة حصنًا قويًا
لكل محب للخير، ومحب لدينه ووطنه.

Translate from Arabic

8:09 AM - 23 Sep 2017

340,776 Retweets 170,760 Likes

Mustafa Hosny
@MustafaHosny

الله يغفر لنا في هذه الليلة لأسباب كثيرة منها تجهيز قلوبنا
لاستقبال أنواره في رمضان..
#ليلة_النصف_من_شعبان

Translate Tweet

Figure 3.1: Overview of the proposed tracking system architecture

3.1 Sentiment Analysis Model

In order to enhance the performance of our sentiment analysis model, we opted to fine-tune five distinct Arabic BERT models: AraBERT, MARBERT, CaMel-bert-DA, CaMel-bert-Mix, and Alanzi. These models possess impressive language comprehension capabilities, making them ideal candidates for our task. In this section, we will delve into the fundamentals of AraBERT's input representations and discuss the process of fine-tuning it to meet our specific needs. By fine-tuning AraBERT, we can adapt it to the particular intricacies of our dataset and achieve more precise outcomes, this process is similar to that of the other models.

The input representation is a crucial aspect of the AraBERT model, as it determines how the model interprets and processes the input text.

AraBERT also requires the input sequence to follow a certain format. The sequence should have a maximum length of 512, with the first token being [CLS], and [SEP] token used to separate sentences. Similar to BERT, the final hidden state of the [CLS] token is used as the overall sequence representation for classification purposes [38]. Since the lengths of sentences can vary within a dataset, ARABERT also works with fixed-length sequences and thus requires padding or truncation. To ensure all sentences have the same length, the special [PAD] token is added.

When using AraBERT tokenization, words with the same meaning but different forms can create redundancy in the vocabulary. For example, the definite article "Al" - "ال" is always prefixed to other words in Arabic but is not a part of the intrinsic meaning of those words. To address this issue, the researchers segmented the words using Farasa into stems, prefixes, and suffixes [3]. They then trained a SentencePiece in unigram mode on the segmented pre-training dataset to create a subword vocabulary of around 60K tokens. This approach allowed them to avoid unnecessary redundancy in the vocabulary. They also created a second version of AraBERT (AraBERTv0.1) that does not require any segmentation by training SentencePiece on non-segmented text. The final vocabulary size was 64k tokens, including nearly 4K unused tokens for further pre-training if necessary [18].

Each token is then mapped to a corresponding vector using an embedding matrix, which assigns a high-dimensional vector representation to each token in the vocabulary. These vectors are learned during the training process and capture the semantic meaning of the tokens in the context of the training data.

Finally, the input text is represented as a sequence of token embeddings, which are fed into the neural network for further processing. This input representation allows Arabert to effectively capture the nuances of the Arabic language and make accurate predictions in various natural language processing tasks, including sentiment analysis.

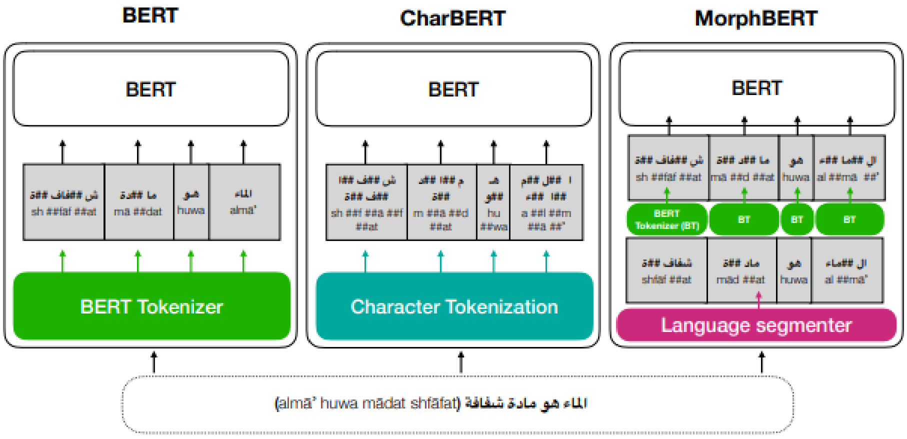


Figure 3.2: Overview of the different Bert Segmentation and Tokenization Techniques. Adapted From : [13]

3.2 Pre-training

BERT Google launched a multilingual BERT model (Devlin et al., 2018) that can handle more than 100 languages with good results in most of them. But it was found that pre-training monolingual BERT models for non-English languages, such as Alberto (Polignano et al., 2019) for Italian and other BERT models that are available to the public (Martin et al., 2019; de Vries et al., 2019), have higher performance than the multilingual BERT model.[18]

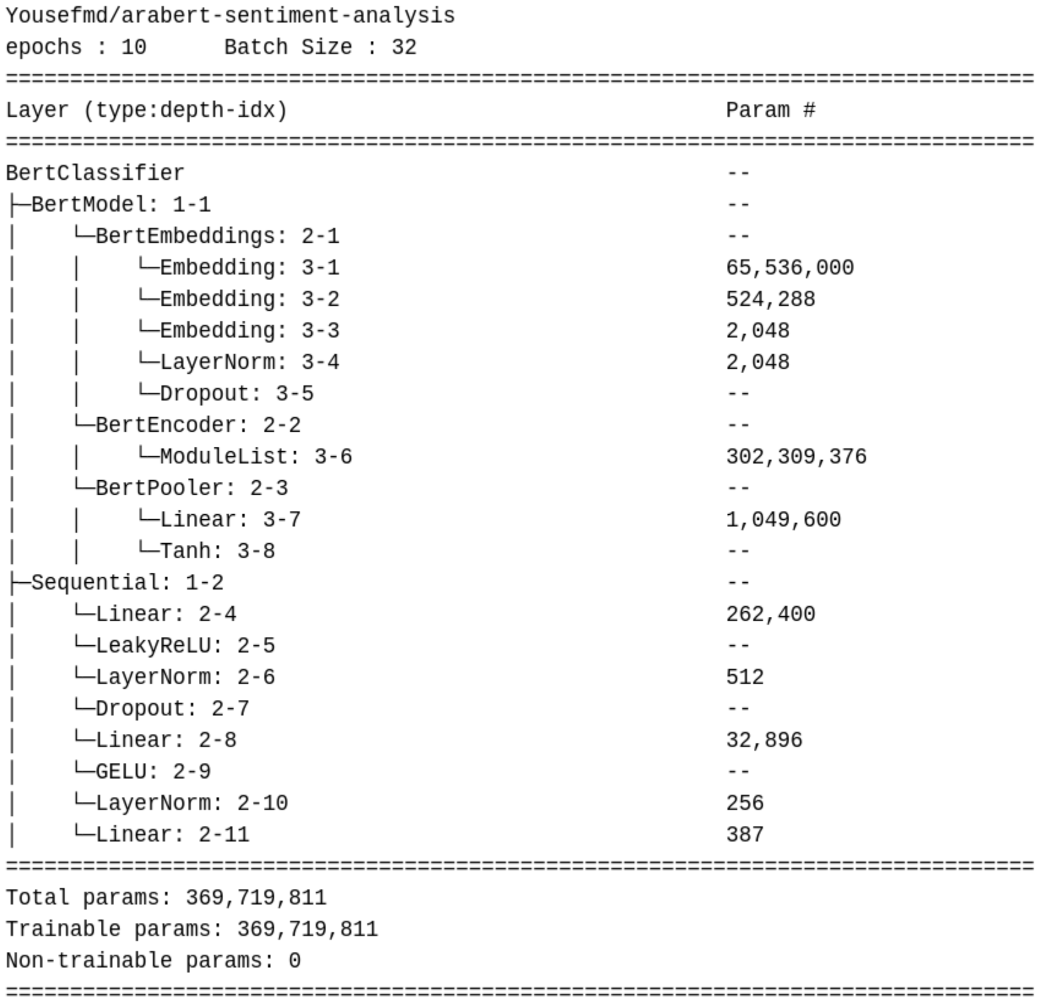
Versatility of BERT During pre-training, BERT learns language by simultaneously training on two unsupervised tasks: the model would learn to predict masked words (i.e., words that are replaced with [MASK] tokens) which is called Masked language modeling (MLM) task which enables the bidirectional context learning from text by masking (hiding) a word in a sentence and forcing BERT to use the words on either side of the covered word to predict it. Next Sentence A prediction (NSP) task was employed to distinguish between two sentences which helps the model understand the relationship between the two sentences. After pre-training, you can fine-tune the model on downstream tasks such as sentiment analysis or named entity recognition.

The reason why it is possible to train BERT in Arabic is that BERT is a language-agnostic model that can be trained in any language for which there is enough data. Arabic is a rich and complex language, but it has a large amount of text available on the web, making it feasible to train BERT in Arabic. Additionally, the success of pre-trained models like AraBERT and mBERT-Base Arabic suggests that training BERT in Arabic can yield good results.

To train a BERT model in Arabic, you would need a large corpus of Arabic text for pre-training. This corpus should be diverse and representative of the language, covering different genres, styles, and topics. You would also need to preprocess the text to tokenize it into subwords, a process known as WordPiece tokenization, which is used by BERT and other transformer models [18].

AraBERT [2] According to Antoun and colleagues [18] , AraBERT employed a significantly larger dataset of approximately 24GB compared to the 4.3GB used for the multilingual BERT, which was trained on Wikipedia. Additionally, the vocabulary size for AraBERT was 64k tokens, which is considerably larger than the 2k tokens used for multilingual BERT. Finally, the larger dataset used for AraBERT resulted in a pre-training distribution with greater diversity.

[2] AraBERT : https://huggingface.co/aubmindlab/bert-base-arabert



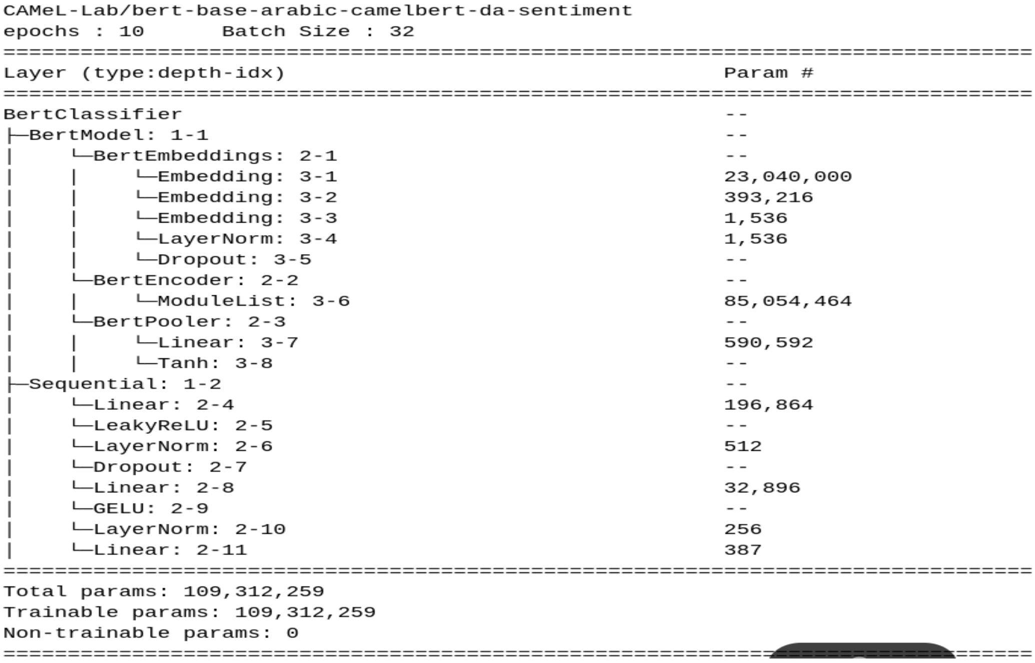
```
Yousefmd/arabert-sentiment-analysis
epochs : 10      Batch Size : 32
================================================================================
Layer (type:depth-idx)                                        Param #
================================================================================
BertClassifier                                                --
├─BertModel: 1-1                                              --
|    └─BertEmbeddings: 2-1                                    --
|    |    └─Embedding: 3-1                                    65,536,000
|    |    └─Embedding: 3-2                                    524,288
|    |    └─Embedding: 3-3                                    2,048
|    |    └─LayerNorm: 3-4                                    2,048
|    |    └─Dropout: 3-5                                      --
|    └─BertEncoder: 2-2                                       --
|    |    └─ModuleList: 3-6                                   302,309,376
|    └─BertPooler: 2-3                                        --
|    |    └─Linear: 3-7                                       1,049,600
|    |    └─Tanh: 3-8                                         --
├─Sequential: 1-2                                             --
|    └─Linear: 2-4                                            262,400
|    └─LeakyReLU: 2-5                                         --
|    └─LayerNorm: 2-6                                         512
|    └─Dropout: 2-7                                           --
|    └─Linear: 2-8                                            32,896
|    └─GELU: 2-9                                              --
|    └─LayerNorm: 2-10                                        256
|    └─Linear: 2-11                                           387
================================================================================
Total params: 369,719,811
Trainable params: 369,719,811
Non-trainable params: 0
================================================================================
```

Figure 3.3: Snapshot of the Overview of the Arabert Architecture and Classification Layer

CAMeLBERT , the researchers behind CAMeLBERT [37] developed a pre-trained language model for Arabic and applied it to three variants of the language: Modern Standard Arabic (MSA), dialectal Arabic [3], classical Arabic and mix of all[4]. They then evaluated the effectiveness of their model by testing it on 12 datasets across five different tasks, including Named Entity Recognition, POS tagging, sentiment analysis, dialect identification, and poetry classification.

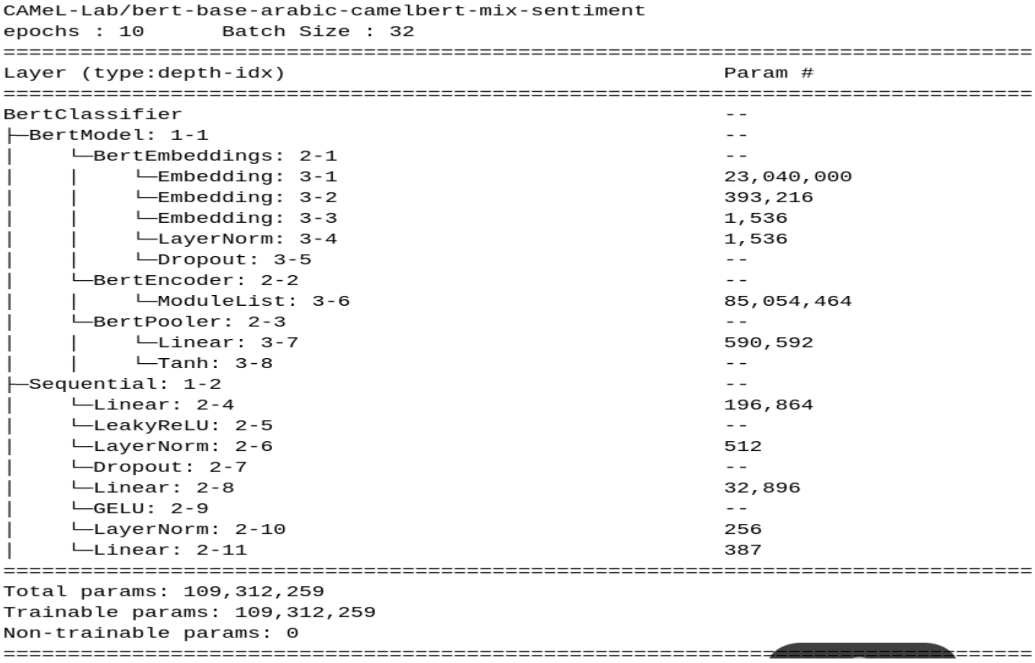
[3] CAMeLBERT-DA:https://huggingface.co/CAMeL-Lab/bert-base-arabic-camelbert-da-sentiment

[4] CaMeLBERT-MIX:https://huggingface.co/CAMeL-Lab/bert-base-arabic-camelbert-mix-sentiment



```
CAMeL-Lab/bert-base-arabic-camelbert-da-sentiment
epochs : 10      Batch Size : 32
================================================================================
Layer (type:depth-idx)                                  Param #
================================================================================
BertClassifier                                          --
├─BertModel: 1-1                                        --
│    └─BertEmbeddings: 2-1                              --
│    │    └─Embedding: 3-1                              23,040,000
│    │    └─Embedding: 3-2                              393,216
│    │    └─Embedding: 3-3                              1,536
│    │    └─LayerNorm: 3-4                              1,536
│    │    └─Dropout: 3-5                                --
│    └─BertEncoder: 2-2                                 --
│    │    └─ModuleList: 3-6                             85,054,464
│    └─BertPooler: 2-3                                  --
│    │    └─Linear: 3-7                                 590,592
│    │    └─Tanh: 3-8                                   --
├─Sequential: 1-2                                       --
│    └─Linear: 2-4                                      196,864
│    └─LeakyReLU: 2-5                                   --
│    └─LayerNorm: 2-6                                   512
│    └─Dropout: 2-7                                     --
│    └─Linear: 2-8                                      32,896
│    └─GELU: 2-9                                        --
│    └─LayerNorm: 2-10                                  256
│    └─Linear: 2-11                                     387
================================================================================
Total params: 109,312,259
Trainable params: 109,312,259
Non-trainable params: 0
================================================================================
```

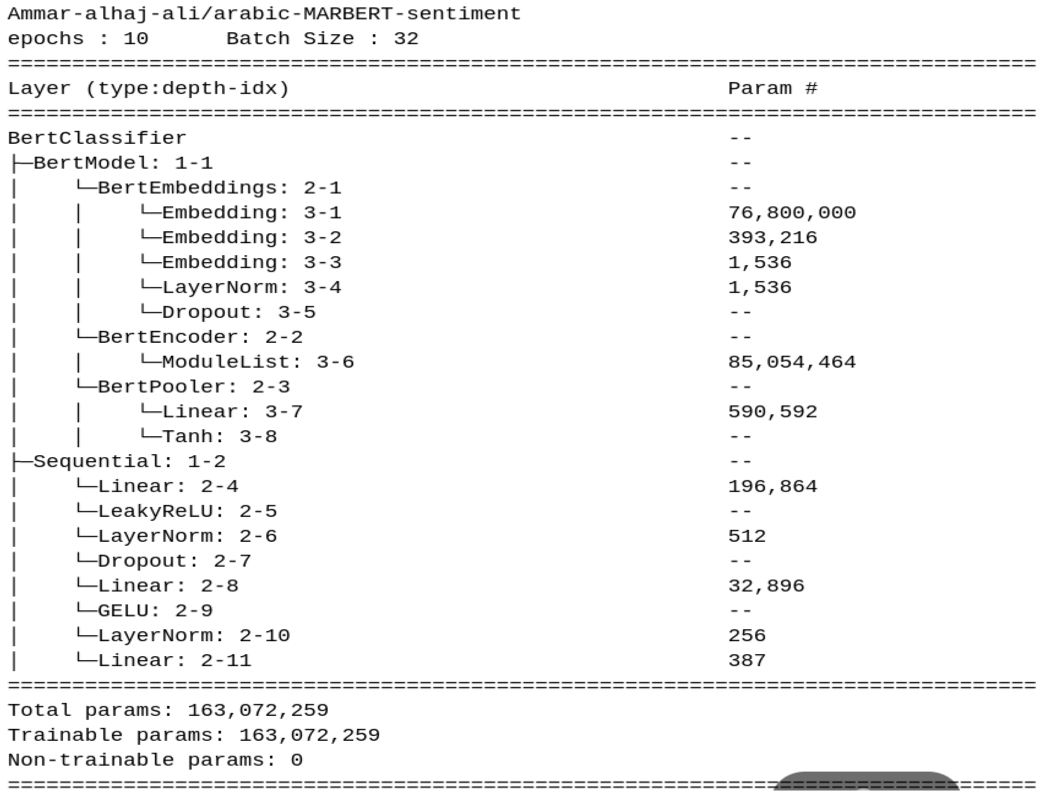
Figure 3.4: Snapshot of the Overview of the CAMel-DA Architecture and Classification Layer



```
CAMeL-Lab/bert-base-arabic-camelbert-mix-sentiment
epochs : 10      Batch Size : 32
================================================================================
Layer (type:depth-idx)                                  Param #
================================================================================
BertClassifier                                          --
├─BertModel: 1-1                                        --
│    └─BertEmbeddings: 2-1                              --
│    │    └─Embedding: 3-1                              23,040,000
│    │    └─Embedding: 3-2                              393,216
│    │    └─Embedding: 3-3                              1,536
│    │    └─LayerNorm: 3-4                              1,536
│    │    └─Dropout: 3-5                                --
│    └─BertEncoder: 2-2                                 --
│    │    └─ModuleList: 3-6                             85,054,464
│    └─BertPooler: 2-3                                  --
│    │    └─Linear: 3-7                                 590,592
│    │    └─Tanh: 3-8                                   --
├─Sequential: 1-2                                       --
│    └─Linear: 2-4                                      196,864
│    └─LeakyReLU: 2-5                                   --
│    └─LayerNorm: 2-6                                   512
│    └─Dropout: 2-7                                     --
│    └─Linear: 2-8                                      32,896
│    └─GELU: 2-9                                        --
│    └─LayerNorm: 2-10                                  256
│    └─Linear: 2-11                                     387
================================================================================
Total params: 109,312,259
Trainable params: 109,312,259
Non-trainable params: 0
================================================================================
```

Figure 3.5: Snapshot of the Overview of the CAMel-MiX Architecture and Classification Layer

MARBERT [5]is a pre-trained language model designed for both Dialectal Arabic (DA) and Modern Standard Arabic (MSA) at a large scale. Given that Arabic has numerous variations, MARBERT was trained by randomly selecting 1 billion Arabic tweets from a large in-house dataset consisting of approximately 6 billion tweets. To be included in the dataset, a tweet had to have at least 3 Arabic words, regardless of whether or not it had non-Arabic characters. This means that non-Arabic characters were not removed as long as the tweet met the 3 Arabic word requirements. The dataset contains 128GB of text, which equates to 15.6 billion tokens. MARBERT utilizes the same network architecture as ARBERT (BERT-base), but without the next sentence prediction (NSP) objective due to the brevity of tweets [4].



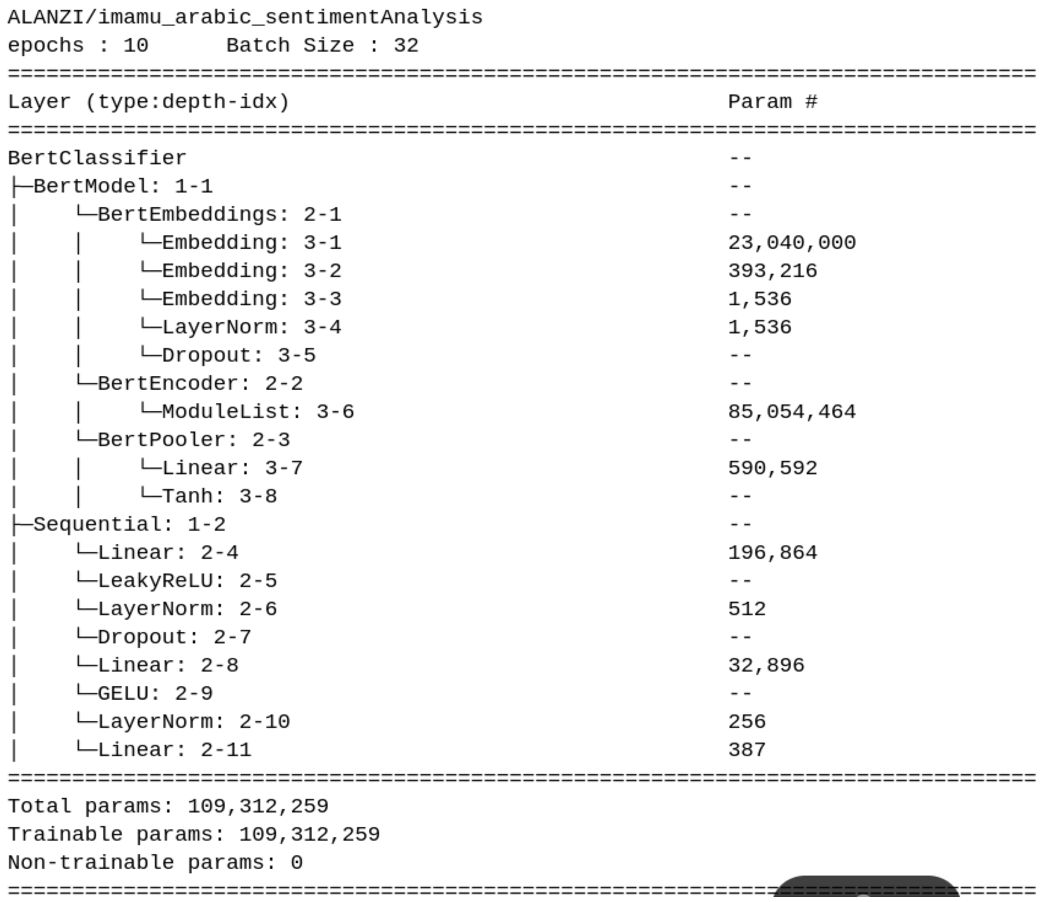
```
Ammar-alhaj-ali/arabic-MARBERT-sentiment
epochs : 10      Batch Size : 32
================================================================================
Layer (type:depth-idx)                                  Param #
================================================================================
BertClassifier                                          --
├─BertModel: 1-1                                        --
│    └─BertEmbeddings: 2-1                              --
│    │    └─Embedding: 3-1                              76,800,000
│    │    └─Embedding: 3-2                              393,216
│    │    └─Embedding: 3-3                              1,536
│    │    └─LayerNorm: 3-4                              1,536
│    │    └─Dropout: 3-5                                --
│    └─BertEncoder: 2-2                                 --
│    │    └─ModuleList: 3-6                             85,054,464
│    └─BertPooler: 2-3                                  --
│    │    └─Linear: 3-7                                 590,592
│    │    └─Tanh: 3-8                                   --
├─Sequential: 1-2                                       --
│    └─Linear: 2-4                                      196,864
│    └─LeakyReLU: 2-5                                   --
│    └─LayerNorm: 2-6                                   512
│    └─Dropout: 2-7                                     --
│    └─Linear: 2-8                                      32,896
│    └─GELU: 2-9                                        --
│    └─LayerNorm: 2-10                                  256
│    └─Linear: 2-11                                     387
================================================================================
Total params: 163,072,259
Trainable params: 163,072,259
Non-trainable params: 0
================================================================================
```

Figure 3.6: Snapshot of the Overview of the MARBERT Architecture and Classification Layer

ALANZI [6] is a pre-trained model that has already been fine-tuned for sentiment analysis was trained on a large corpus of Arabic language data. It was trained on 30,646 sentences for each label and achieved an accuracy score of 84%.

[5] MARBERT : https://huggingface.co/ALANZI/imamu$_a$rabic$_s$entimentAnalysis

[6] Alanzi : https://huggingface.co/ALANZI/imamu$_a$rabic$_s$entimentAnalysis



```
ALANZI/imamu_arabic_sentimentAnalysis
epochs : 10      Batch Size : 32
================================================================================
Layer (type:depth-idx)                                  Param #
================================================================================
BertClassifier                                          --
├─BertModel: 1-1                                        --
│    └─BertEmbeddings: 2-1                              --
│    │    └─Embedding: 3-1                              23,040,000
│    │    └─Embedding: 3-2                              393,216
│    │    └─Embedding: 3-3                              1,536
│    │    └─LayerNorm: 3-4                              1,536
│    │    └─Dropout: 3-5                                --
│    └─BertEncoder: 2-2                                 --
│    │    └─ModuleList: 3-6                             85,054,464
│    └─BertPooler: 2-3                                  --
│    │    └─Linear: 3-7                                 590,592
│    │    └─Tanh: 3-8                                   --
├─Sequential: 1-2                                       --
│    └─Linear: 2-4                                      196,864
│    └─LeakyReLU: 2-5                                   --
│    └─LayerNorm: 2-6                                   512
│    └─Dropout: 2-7                                     --
│    └─Linear: 2-8                                      32,896
│    └─GELU: 2-9                                        --
│    └─LayerNorm: 2-10                                  256
│    └─Linear: 2-11                                     387
================================================================================
Total params: 109,312,259
Trainable params: 109,312,259
Non-trainable params: 0
================================================================================
```

Figure 3.7: Snapshot of the Overview of the Alanzi Architecture and Classification Layer

3.3 Fine-tuning

After pre-training, there are two approaches to fine-tuning BERT generally. The first approach is known as ”feature extraction”, in which we utilize the final output of pre-trained BERT as input to another model while keeping the BERT architecture intact. This allows us to extract features to use as contextualized word embeddings, which can then be used as input to a separate model for the actual task.

The second approach is ”fine-tuning” BERT, which involves modifying the architecture by adding an additional layer(s) on top of BERT and then training the whole model together. With this method, the additional layer(s) are trained from scratch, but the BERT weights are only fine-tuned, making training time shorter[17] .

The fine-tuning process aims to teach the pre-trained model to learn representations that are specifically useful for a given task. By fine-tuning the model, we can improve its performance on the task at hand and obtain better results.

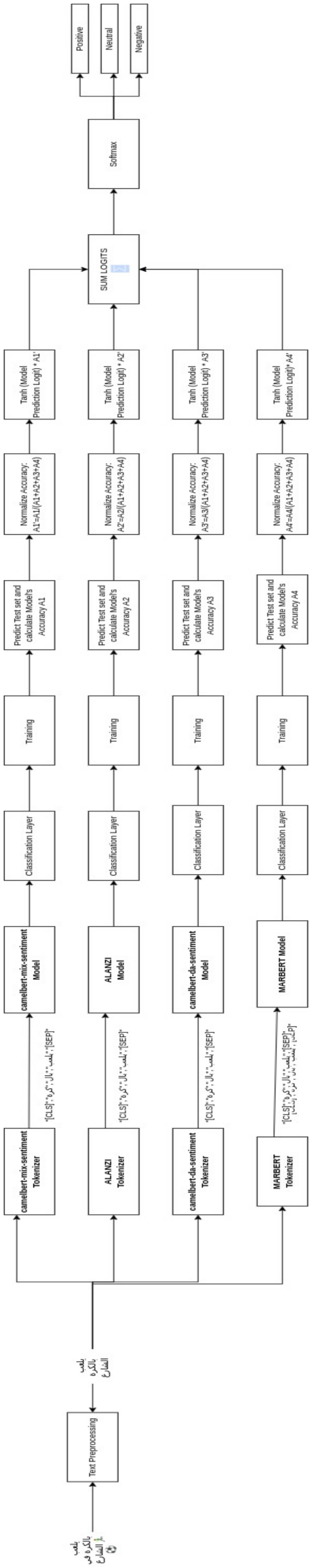

يلعب بالكره فى الشارع
Text Preprocessing
يلعب بالكره الشارع
camelbert-mix-sentiment Tokenizer
ALANZI Tokenizer
camelbert-da-sentiment Tokenizer
MARBERT Tokenizer
camelbert-mix-sentiment Model
ALANZI Model
camelbert-da-sentiment Model
MARBERT Model
Classification Layer
Training
Predict Test set and calculate Model's Accuracy A1
Predict Test set and calculate Model's Accuracy A2
Predict Test set and calculate Model's Accuracy A3
Predict Test set and calculate Model's Accuracy A4
Normalize Accuracy: A1'=A1/(A1+A2+A3+A4)
Normalize Accuracy: A2'=A2/(A1+A2+A3+A4)
Normalize Accuracy: A3'=A3/(A1+A2+A3+A4)
Normalize Accuracy: A4'=A4/(A1+A2+A3+A4)
Tanh (Model Prediction Logit) * A1'
Tanh (Model Prediction Logit) * A2'
Tanh (Model Prediction Logit) * A3'
Tanh (Model Prediction Logit)* A4'
SUM LOGITS
Softmax
Positive
Neutral
Negative

Figure 3.8: Overview of the sentiment analysis model architecture and plan

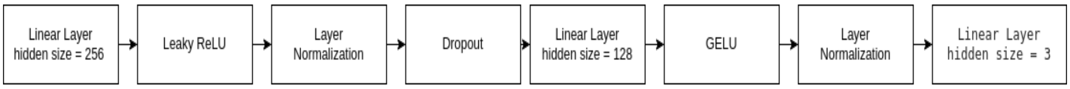


Figure 3.9: Classification Layer

In this Thesis, we decide to use the fine-tuning approach with 5 different Arabic BERT models for the sentiment analysis task by adding on top of each model a classification FFN layer but we will see in Experimentation that AraBERT is under-fitting so we are going to discard it.

Classification FFN layer is composed of a linear layer with a hidden size of 256, followed by an activation function called Leaky ReLU. After that, it passes through Layer normalization, which standardizes the inputs of a layer by normalizing the mean and variance of the activations (as explained in this Section 4.6.1). Then, a Dropout layer is applied with a rate of 0.4 to enable generalization. Another Linear Layer with a hidden size of 128 follows, which passes through an activation function called GELU. After that, it goes through another Layer of normalization and finally reaches the linear layer of hidden size 3, which predicts each label. As depicted in this Figure 3.9

3.4 Sentiment Forecasting

We collected an English Tweets timeline dataset[7] and converted it into a time-indexed monthly format. The dataset was cleaned and translated into Arabic using the Helsinki-NLP/opus-mt-en-ar translation model[8] from Hugging Face. The translated tweets were then preprocessed using subword tokenization with the Camel-BERT tokenizer.

We used our Arabic Ensemble Model that we introduced in this previous Section 3.1 to predict the sentiment of the preprocessed tweets. To train our LSTM model, we used a sliding window of 10, meaning the input included tweets from X_{t-10} to X_{t-1}, and the output was the sentiment of time X_t. We split the dataset into three sets with an 80:10:10 ratio for training, validation, and testing, respectively.

Afterward, we trained the LSTM model[9] on the train set using the Keras deep learning library with the following layers: LSTM Layer with 64 hidden units, the Dense layer with 8 hidden units, and the activation function relu. The model was then trained on the classification Dense layer with 3 hidden units and the softmax activation function.

Finally, we evaluated the performance of the LSTM model on our test set. We measured the accuracy, precision, recall, and F1-score of the model, and compared its performance to other sentiment analysis models for Arabic tweets. We also performed a sensitivity analysis to assess the impact of hyperparameters on the model's performance and compared it with the Actual Sentiment time series through line plots.

[7] Dataset :https://www.kaggle.com/datasets/zoegreenslade/twittermhcampaignsentmentanalysis

[8] Translation Model: https://huggingface.co/Helsinki-NLP/opus-mt-ar-en

[9] Notebook : https://colab.research.google.com/drive/1SJKG2zJagVmVA9IQwGFBiZqsxlbXT29F?usp=sharing

3.5 Topic Modeling

We conducted a comparison between BERTopic and two of the most widely used topic modeling techniques, namely LDA and NMF. Since LDA and NMF models necessitate the pre-specification of the number of topics, we opted to initiate our experiment with five topics and we trained the models on this Preprocessed Dataset (as explained in this Section 4.3).

BERTopic is a topic modeling algorithm that employs a pre-trained language model, such as BERT, to generate embeddings for the text data. It subsequently applies a clustering algorithm to group the data into topics. By leveraging a pre-trained language model, BERTopic can capture the semantic meaning of the text data and generate high-quality topics that are both coherent and interpretable.

Afterward, we aimed to examine the average sentiment score of documents assigned to each topic, which is referred to as Aspect topic modeling.

Aspect topic modeling typically involves a combination of techniques such as text preprocessing, topic modeling, and sentiment analysis. The goal is to identify specific aspects or features of a product, service, or domain that are frequently mentioned in customer feedback and understand the sentiment associated with each aspect[49].

Chapter 4

Experimentation & Results

We conducted a series of experiments and analyses on our models, aiming to delve into their capabilities and performance. In this study, we explored various methodologies and techniques to enhance the accuracy and efficiency of our models. Through rigorous testing and evaluation, we obtained insightful results and valuable insights, shedding light on the potential of our NLP models for sentiment analysis, forecasting, and Topic Modeling tasks.

4.1 Experiments Setup

Data curation was conducted on a personal computer with a processing speed of 2.80 GHz and a RAM capacity of 16 GB. All other experiments were carried out on Google Colab notebooks, utilizing an NVIDIA Tesla T4 GPU with a High-RAM of 25.46 GB.

4.2 Data Acquistion

As a result of the limited availability of Egyptian dialect Arabic and Modern Standard Arabic (MSA) tweets, as well as their diverse range of topics, I have conducted thorough investigations of each dataset individually. Subsequently, I have merged them to create a comprehensive balanced dataset that can be utilized in its entirety for subsequent stages in the pipeline, such as sentiment analysis tasks. This approach ensures a robust and representative dataset for further analysis and enhances the accuracy and reliability of the results.

- ASTD Dataset [1]: This a set of Egyptian-dialect-Arabic tweets containing over 10,000 entries mostly about News. They are labeled as one of four classes: objective, subjective negative, subjective positive, and subjective mixed. we ignored subjectively mixed, and the objective was Neutral.

[1] ASTD Dataset : http://www.mohamedaly.info/datasets/astd

- ArSAS Dataset [2]: This a set of Egyptian-dialect-Arabic tweets containing around 20,000 entries that contain different topics regarding Events and Entities. They are labeled as one of four classes: negative, positive, and mixed. we ignored mixed.

Table 4.1: ArSAS Sentiment Counts

Sentiment	Count
Negative	7384
Neutral	6894
Positive	4400
Mixed	1219

- Egy-tweets Dataset[3]: This a set of Egyptian-dialect-Arabic tweets containing around 3500 entries.

Table 4.2: Egy-tweets Sentiment Counts

Sentiment	Count
Negative	1278
Neutral	1168
Positive	987

- Emotional-Tone Dataset [4]: it is an Egyptian Dialect dataset that is labeled by emotions I removed emotions that have mixed sentiments and replaced correct emotions with their respective sentiments after a careful investigation which resulted in 5550 entries.

Table 4.3: ArTwitter Sentiment Counts

Sentiment	Count
Negative	2800
Neutral	1500
Positive	1250

- ArTwitter [5]: This is a set of MSA tweets containing around 600 entries.

[2] ArSAS Dataset:https://huggingface.co/datasets/arbml/ArSAS

[3] Egy-tweets Dataset :https://drive.google.com/file/d/1pA73h2LYfie3wimSvYwJLXz8keiS6bnv/view?usp=share$_{l}ink$

[4] Emotional-Tone Dataset :https://huggingface.co/datasets/emotone$_{a}r$

[5] ArTwitter :https://drive.google.com/file/d/1pA73h2LYfie3wimSvYwJLXz8keiS6bnv/view?usp=share$_{l}ink$

Table 4.4: ArTwitter Sentiment Counts

Sentiment	Count
Negative	228
Neutral	192
Positive	263

- ARSARCASM [6] This is a set of MSA tweets that are sarcastic containing around 12500 entries.

Table 4.5: ARSARCASM Sentiment Counts

Sentiment	Count
Negative	4597
Neutral	5736
Positive	2178

- ar$_{r}eviews_{1}00k$[7]: is a set of tweets written in Modern Standard Arabic (MSA) and Egyptian Dialect Arabic, totaling approximately 66,000 entries. However, after thorough cleaning and investigation, only 32,000 entries were retained due to numerous instances of incorrect sentiment corresponding to the respective sentiment.

Table 4.6: ar$_{r}eviews_{1}00kSentimentCounts$

Sentiment	Count
Negative	16000
Positive	16000

4.3 Text Preprocessing

I have conducted extensive studies on the importance of text preprocessing [8] in achieving accurate and reliable results in sentiment analysis tasks. Text preprocessing plays a crucial role in transforming raw text data into a structured format that can be effectively analyzed and used for our Models. As Depicted in the following Figures 4.1 4.2 how they contain noise, the high variety of vocabulary, and the number of words and characters.

[6] ARSARCASM :https://drive.google.com/file/d/1pA73h2LYfie3wimSvYwJLXz8keiS6bnv/view?usp=share$_link$

[7] ar$_{r}eviews_{1}00k$: $https : //www.kaggle.com/code/abedkhooli/ar - reviews - 100k$

[8] Notebook : https://colab.research.google.com/drive/1AP-xbKQhJsfKsqKKFjfL6faWUHEkkttq?usp=sharing

Figure 4.1: Original Tweets - Word Cloud

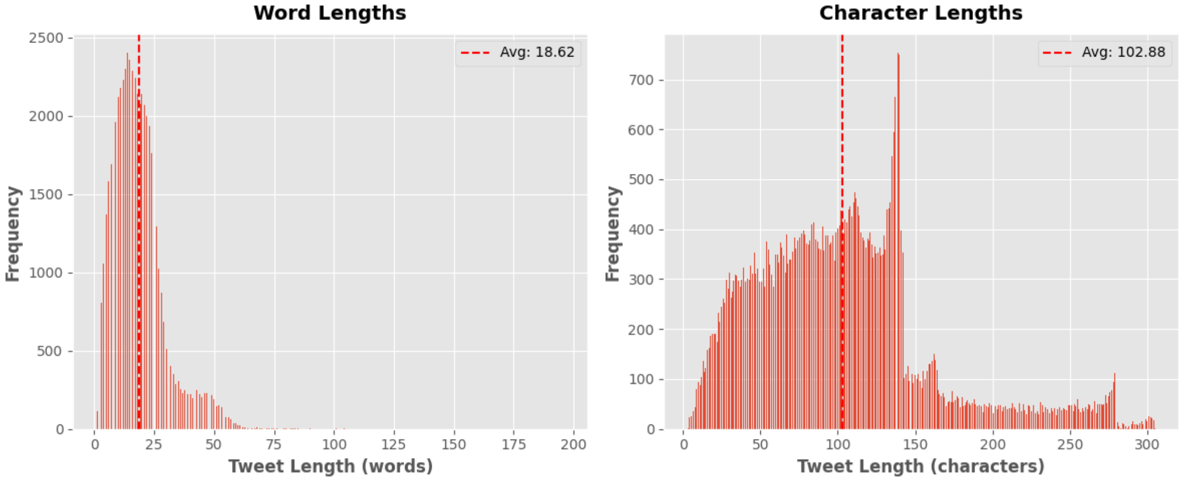
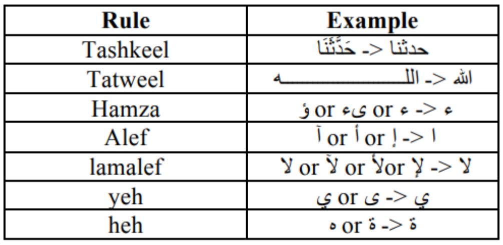


Figure 4.2: Original - Word Character Length

4.3.1 Text Normalization

We opted to use a normalization technique for its consolidating a specific group of letters to their respective letter, as there is limited scope for further improvements in this area. The language normalization rules are outlined in Figure 4.3



Rule	**Example**
Tashkeel	حَدَّثَنَا -> حدثنا
Tatweel	اللـــــــــــــــه -> الله
Hamza	ؤ or ىء or ء -> ء
Alef	آ or أ or إ -> ا
lamalef	لآ or لأ orلإ -> لا
yeh	ي or ى -> ي
heh	ه or ة -> ة

Figure 4.3: Normalization Rules. Adapted From:[16]

4.3.2 Replaced Lexicons with one same Semantic Word

The Egyptian-dialect-Arabic Lexicons Data Set is a valuable resource that contains various linguistic elements such as intensifiers, negators, post-negators, negative and positive lexicons. These lexicons can be in the form of unigrams or bigrams, providing a comprehensive coverage of the Egyptian dialect of Arabic.

In order to streamline and optimize the vocabulary size, a replacement strategy has been implemented. The aim of this strategy is to replace certain words found in the lexicons with semantically equivalent alternatives, thereby reducing the overall number of distinct words in the vocabulary.

The Figure 4.7 depicts the process of this replacement strategy. It showcases the flow of replacing the original words with their semantically equivalent counterparts. By doing so, the data set can maintain its linguistic richness and integrity while effectively reducing the overall vocabulary size.

This replacement approach offers several benefits. Firstly, it helps simplify the data set by collapsing synonymous words into a single representative, thereby avoiding redundancy and repetition. Secondly, it contributes to reducing the computational complexity associated with handling a large number of distinct words, which can improve processing speed and efficiency.

Moreover, this replacement strategy ensures that the semantic nuances and meanings conveyed by the original lexicons are preserved. The selected semantically equivalent words are carefully chosen to maintain the intended expressions, sentiments, and linguistic properties present in the original lexicons.

By reducing the vocabulary size without sacrificing linguistic richness, this replacement strategy enhances the usability and effectiveness of the Egyptian-dialect-Arabic Lexicons Data Set. Researchers and developers can benefit from a more streamlined and efficient resource for various natural language processing tasks, such as sentiment analysis, text classification, and machine translation, among others.

Overall, the process of replacing words in the lexicons with semantically equivalent alternatives demonstrates a thoughtful approach to optimize the data set's vocabulary, making it a valuable asset for studying and understanding the Egyptian dialect of Arabic.

4.3.3 Cleaning Tweet

The identification and removal of non-textual contents and irrelevant information were performed during the analysis process. These included quotation marks (" "), hashtag symbols (), Twitter user mentions (@xxx), URLs, special Unicode and HTML characters like (), Twitter jargon such as "RT" (representing re-tweet) and "QT" (representing quotes), as well as unnecessary white spaces and tabs. To further enhance the analysis, some researchers have recommended normalizing user mentions by replacing them with a standardized term such as مستخدم , urls with رابط ,and emails with ايميل.

Table 4.7: Egyptian Arabic Lexicons TreeBank. Adapted From:[40]

Type	Arabic Lexicons	Word
Intensifiers	آكبيرْ، آكبير جداْ، آقويْ، آجداْ، أويْ، ءويْ، آفشخْ، آلخثْ، آطحنْ، آخالصْ، آبعنفْ، آبقوهْ، آبشدهْ، آبغباوهْ، آتماماْ، ألسنينْ، آلخت ردمْ، آنيك، آخالصْ، آمررهْ، آجداْا، آبشددهْ، آقوويْ	جدا
Negators	آلاْ، آلمْ، آليشْ، آليستْ، آمشْ، آلنْ، آموْ، آموتْ، آغيرْ، آلا يوجدْ، آعدمْ، آمشْ، آليسواْ، آتراجعْ، آمعدومةْ، آمعدومْ، آمفيشْ، آمافيْ، آما فيْ، آمافيشْ، آما فيشْ، آيخلو منْ، آمهوْ، آمبْ، آمافيهْ، آمافيهْ، آمابهْ، آموفيهْ، آمو فيهْ، آما بهْ، آولا يوجد به ايْ، آولاْ، آوغيرْ، آومشْ، آوليشْ، آلا شيءْ، آلا يوجد شئْ، آما اشوف فيه شيْ، آماأشوف فيه شيْ، آوالله مناْ، او الله مناْ	مش
Post-Negators	آمعدومةْ، آمعدومْ، آمتدنيْ، آمتدنيهْ، آغير موجودْ، آغير موجودهْ	غير موجود
Negative Lexicons	„ابقي قابلني„، „ابن الاحي„، „ابن القرعه„، „ابن المره„، „ابن المنحوسة„، „ابن الهرمة„، „ابن جزمه„، „ابو جهل„، „اتق الله„، „اتقو الله„، „اتقي الله„، „محدش هيسال فيا„، „محدش هيسال فيك„، „محدش هيعبرك„، „محدود الذكاء„، „محدوده الذكاء„، „محدودي الذكاء„، „مختل عقليا„، „مخه صغير„، „مخيب للامال„، „مر مرور الكرام„، „مرض اجتماعي„، „مزبله التاريخ„، „مزودها شويه„، „مستواه انخفض„، „مستواه نزل„، „مستواها نزل„، „مسلطين علي„، „مش اوي„، „مش طايق„، „مش طايق نفسه„، „مش طايق نفسي„، „مش طايقاك„، „مش عيب عليك„، „مش عيب عليكم„، „مع الأسف„، „معندكش دم„، „معندكيش دم„، „معندهاش إحساس„، „معندهاش دم„، „معندهاش ضمير„، „معندهمش دم„، „معندهوش إحساس„، „معندهوش دم„، „معندهوش ضمير„، „معندهومش إحساس„، الوسخه„،	وحش
Positive Lexicons	„ذكريات جميله„، „ذوقك عالي„، „راحه البال„، „راضي عن„، „راضي عنك„، „رافع راسنا„، „رب العرش العظيم„، „ربك كريم„، „ربنا ما يحرمني„، „ربنا معاك„، „ربنا معاكم„، „ربنا نفخ في صورته„، „ربنا يبارك„، „ربنا يباركلك„، „ربنا يحمي„، „ربنا يحميك„، „ربنا يديم المحبه„، „ربنا يزيد„، „ربنا يزيد ويبارك„، „ربنا يزيدك„، „ربنا يسعدك„، „ربنا يفتح عليك„،	حلو

4.3.4 Transformation of emojis and emoticons to their respective Arabic Words

transformation is done using pre-defined mappings or dictionaries that associate each emoji or emoticon with its Arabic word equivalent. For example, a smiling face emoji can be transformed into the Arabic word ”سعيد” , as depicted in the following Figure 4.4

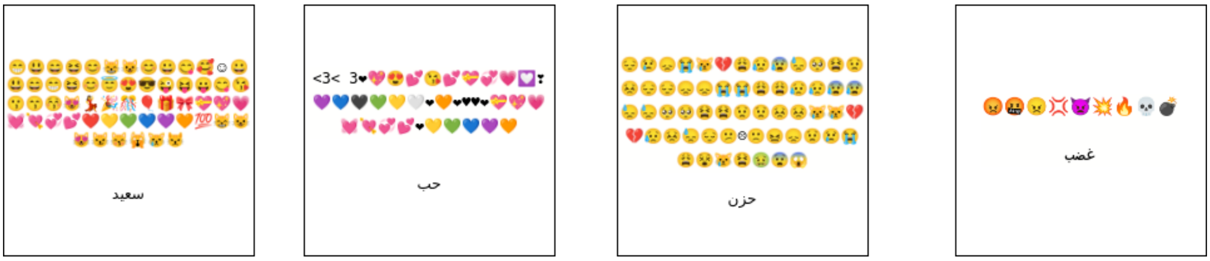


Figure 4.4: Transformation of emojis and emoticons to their respective Arabic Words

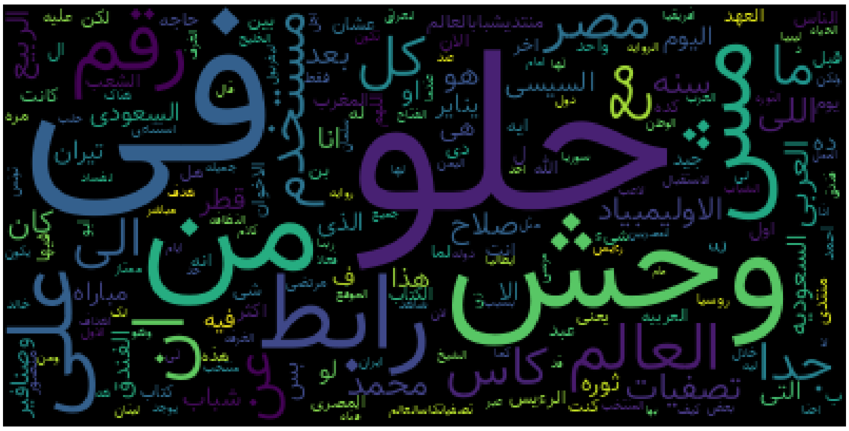


Figure 4.5: After preprocessing - Word Cloud

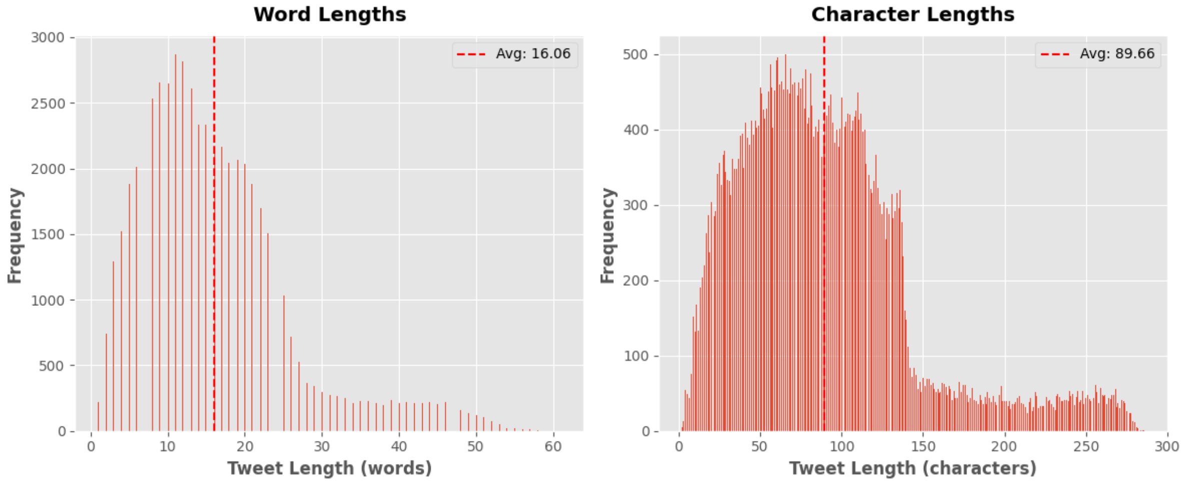


Figure 4.6: After preprocessing - Word Character Length

As we can see in both of these Word Cloud Figures: Original Word Cloud4.1 and Preprocessed Word Cloud 4.6 how different they are from each other replaced links, undescriptive words with their respective unified words, diacritics and more as explained in previous steps.

- Average Word length has decreased from 18.62 to 16.06
- Average Character length has decreased from 102.88 to 89.66
- Vocabulary difference between Original Tweets: 141242 words and Preprocessed Tweets 103433 words: 37809 words
- Changed rows from total rows = 5583 /57000

4.3.5 Stop-words Removal from Tweets

Used a predefined list of 742 stop words from nltk that are commonly used in the Arabic language, such as conjunctions, prepositions, and pronouns. These stop words are often removed from the tweets during the preprocessing stage to reduce the noise in the data and improve the quality of text analysis, as shown in Figure 4.7. The effect of stop words removal can be seen in Figure 4.8.

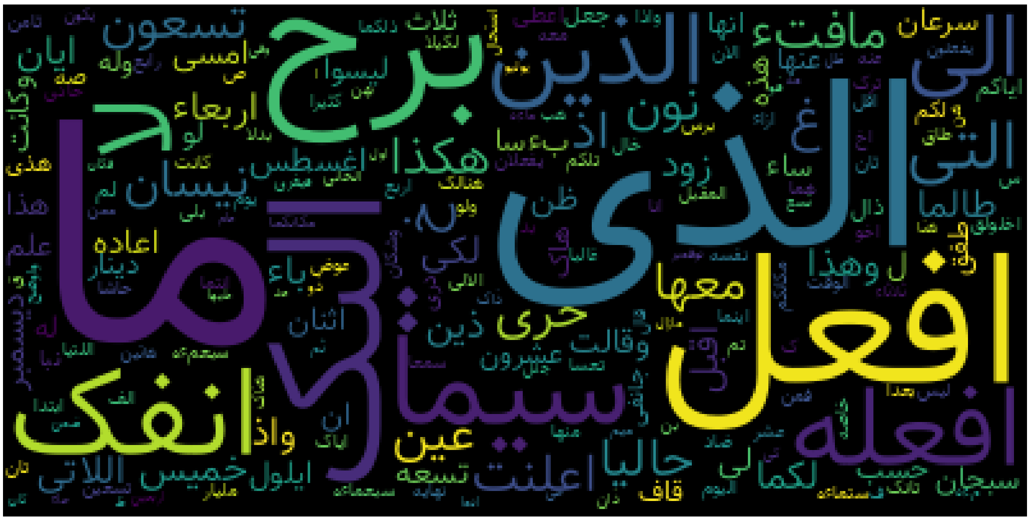


Figure 4.7: Stop Words to Remove - Word Cloud

As you can see in this Word Cloud Figure 4.7 a summary of stopwords in Arabic that does not have semantic meaning nor contributes to the meaning of the sentence.

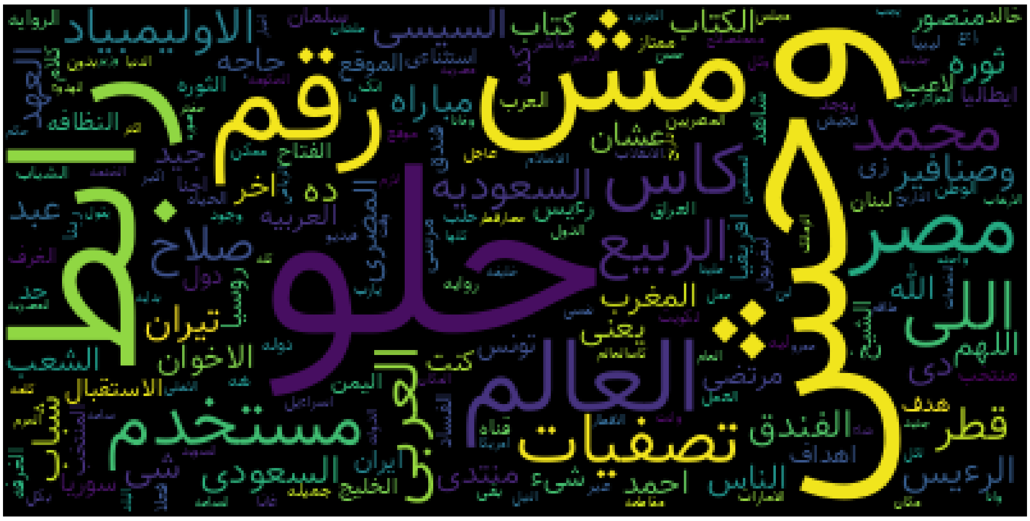


Figure 4.8: After Stop words Removal - Word Cloud

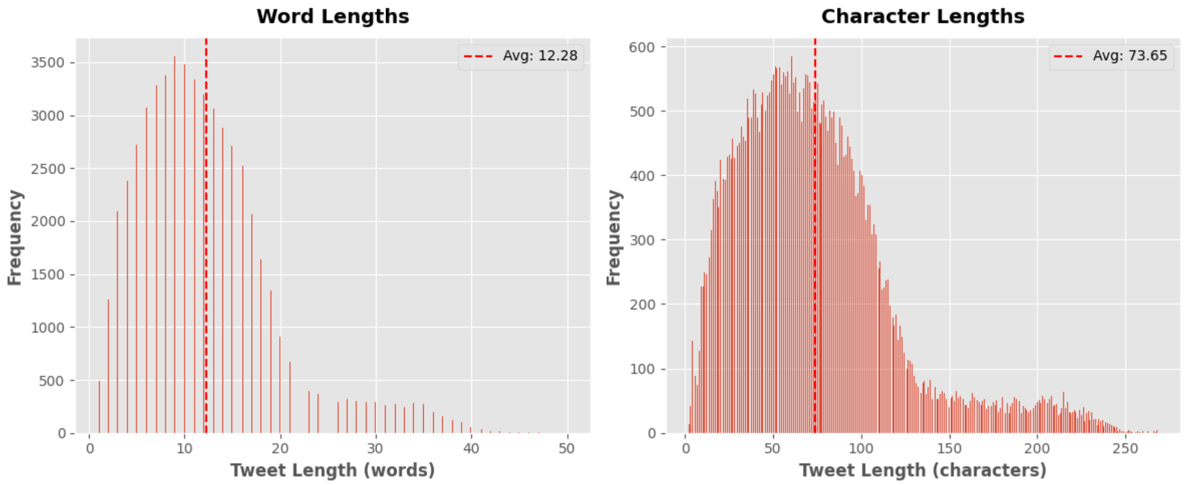


Figure 4.9: After Stop-words Removal - Word Character Length

- Average Word length has decreased from 16.06 to 12.28.

- Average Character length has decreased from 89.66 to 73.65.

- Vocabulary difference between Preprocessed Tweet: 103433 words and After Stop-words Removal from Preprocessed Tweet 98438 words: 4995 words.

- Changed rows from Total rows = 51114 /57000 .

4.3.6 Stemming the Normalized Tweets

We have implemented a light-stemming approach that utilizes a set of dialect-specific prefixes and suffixes. This choice was made to avoid aggressive stemming techniques that reduce words to their roots, as this can often lead to multiple related terms with distinct meanings being mapped to a single root.

The implemented stemmer consists mainly of 4 stages:1) prefix removal, 2) suffix removal, and 3) infix removal 4) Light Stemmer [3],which is mainly applying the rules for broken plurals. Generally, the prefix removal, followed by the suffix removal stage, the infix removal stage, and finally applying a LightStemmer on words that are unchanged and whose length is bigger than 3, Stemming Rules are outlined in this study [16].

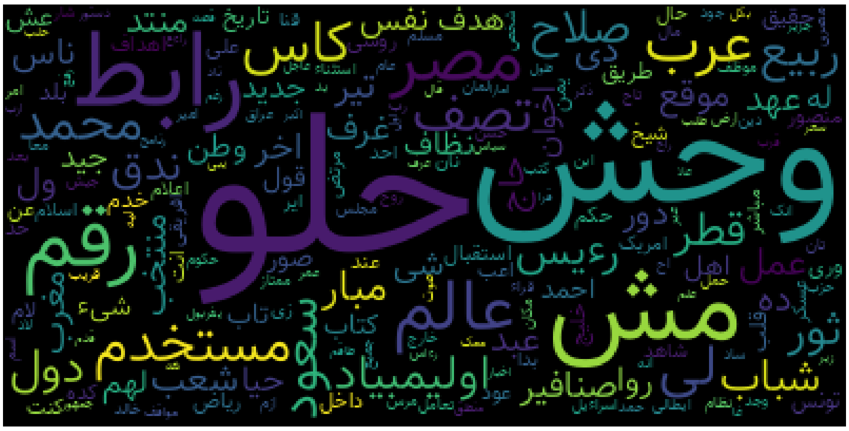


Figure 4.10: After Stemming - Word Cloud

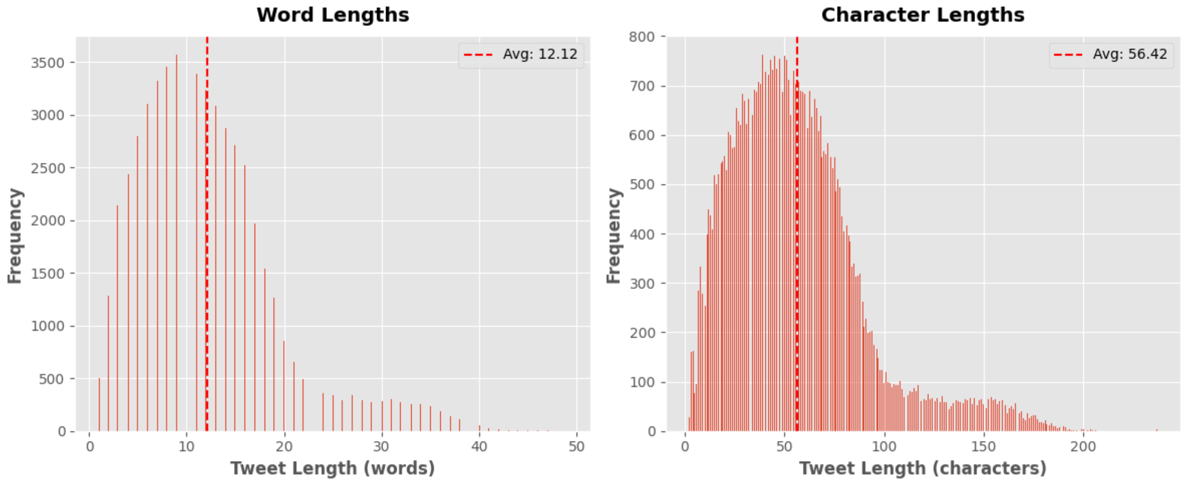


Figure 4.11: After Stemming - Word Character Length

- Average Word length has decreased from 12.28 to 12.12
- Average Character length has decreased from 73.65 to 56.42
- Vocabulary difference between After Stopword removal from Preprocessed Tweet: 98438 words and After Stemming & Stopword removal from Preprocessed Tweet 42446 word: 55992.
- Changed rows from Total rows = 56671 /57000

4.4 Investigating Data Correctness

Data correctness is a crucial aspect in natural language processing (NLP) tasks, as the accuracy and reliability of the data greatly impact the performance and validity of NLP models. Investigating data correctness involves examining the quality and consistency of the data used for training. This process typically includes verifying the accuracy of annotations, assessing data biases, identifying potential errors or inconsistencies, and implementing measures to address any identified issues.

4.4.1 Term Frequency based on Sentiment

Term frequency based on sentiment is a method of analyzing text data to determine the frequency of occurrence of specific words within sentiment-labeled data. This method can be useful for understanding how sentiment is expressed in the language used, as well as for identifying patterns or trends in the data. By analyzing the frequency of terms, we can determine whether these frequent terms are associated with their respective sentiments or not.

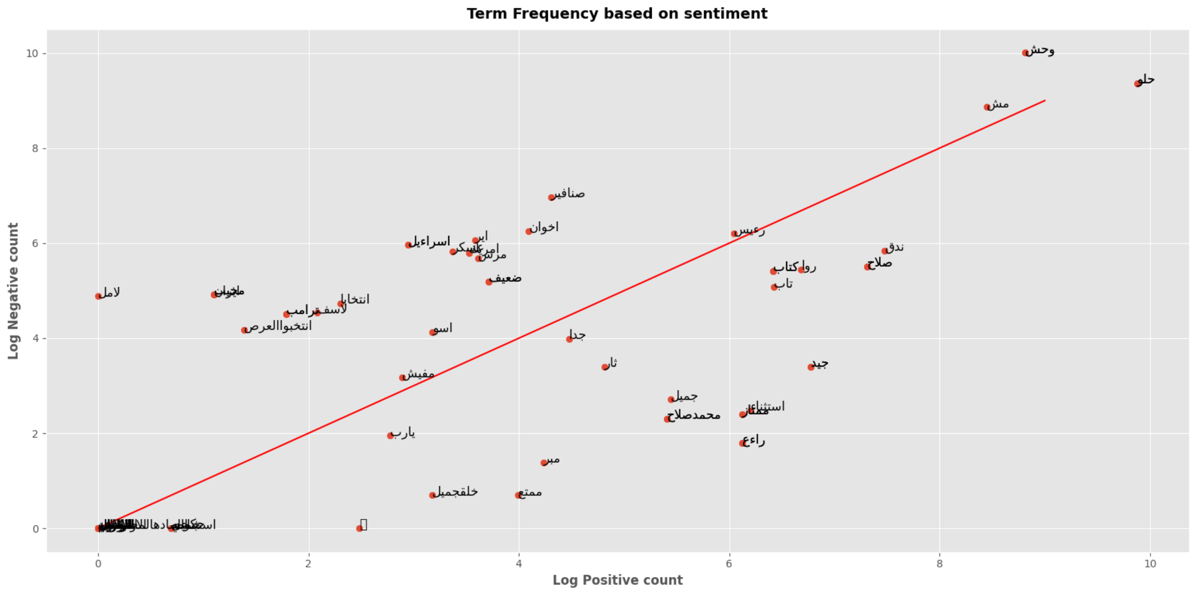


Figure 4.12: Term Frequency based on Sentiment

As shown in Figure 4.12 clearly demonstrates a strong correlation between the meanings and semantics of the terms used in the sentiment analysis task and their respective sentiments.

4.4.2 WSS and Elbow Method

The within-cluster sum of squares (WSS) is a metric used in clustering algorithms, such as k-means, to measure the homogeneity or compactness of the clusters, it is defined as the sum of the squared distances between each data point in a cluster and the cluster's centroid. The elbow method involves plotting the WSS as a function of the number of clusters and selecting the number of clusters where the rate of decrease in WSS starts to level off, as this point represents the best balance between cluster homogeneity and cluster quantity. However, it can also be used to help with dataset correctness in sentiment analysis tasks. By helping us identify any potential inconsistencies in the dataset labeling, as clusters that are not closely aligned with the actual sentiment labels may indicate an issue with the dataset.

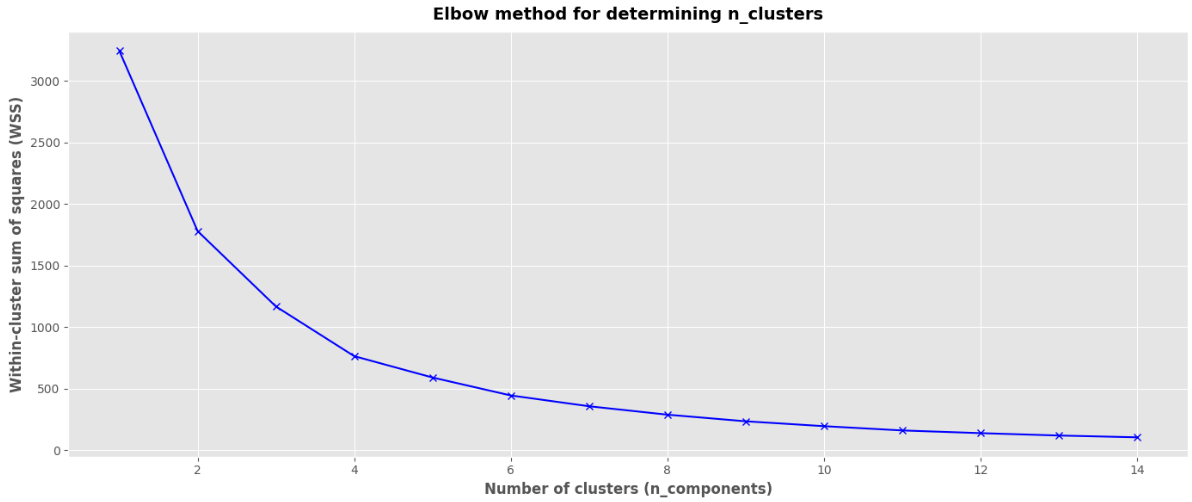


Figure 4.13: Elbow method for determining n$_{c}lusters$

As shown in Figure 4.13, the point at which the rate of decrease in within-cluster sum of squares (WSS) starts to level off is at 5 clusters. We can assume that these 5 clusters represent very negative, negative, neutral, positive, and very positive sentiments. Therefore, we can conclude that the data is correctly labeled.

4.5 Data Preparation

Data preparation plays a crucial role in Natural Language Processing (NLP) tasks. Effective data preparation techniques ensure that the data is suitable for analysis and modeling.These techniques help in improving the quality of the data, reducing noise, and extracting meaningful features for subsequent analysis and modeling.

4.5.1 Spelling Mistakes

In order to focus on the overall sentiment of a tweet, we have decided to overlook misspelled words. As incoming tweets are often not perfectly written, we must account for this in our training data to be able to generalize well. While this approach may not be perfect, it is suitable because misspelled words do not significantly impact the accuracy of sentiment analysis results.

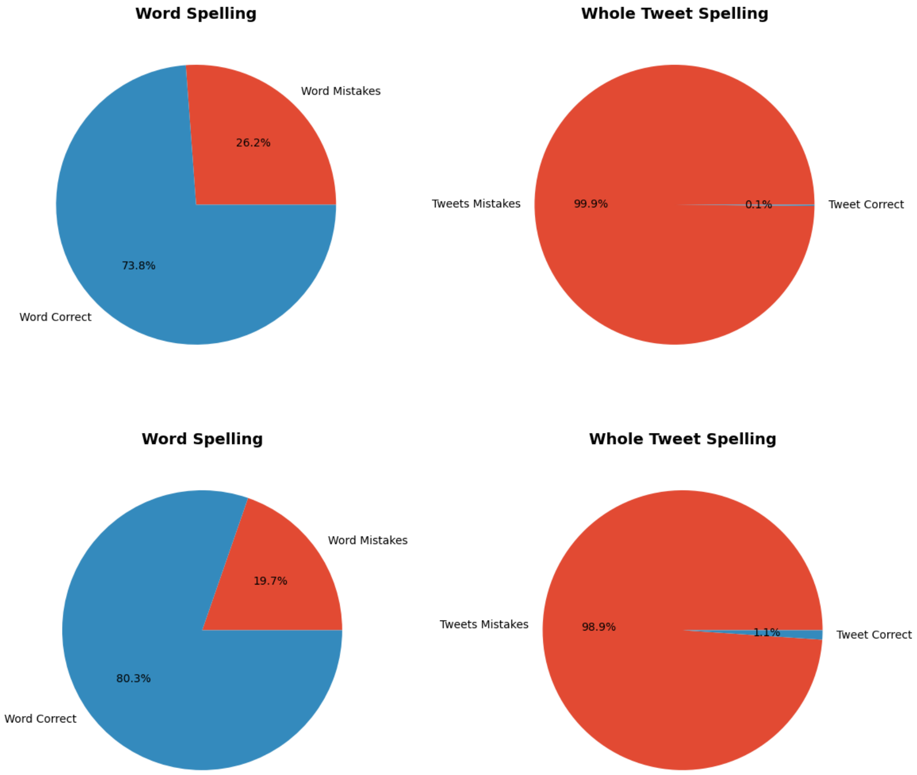


Figure 4.14: Comparison of original tweets and preprocessed tweets.

Surprisingly, our text preprocessing steps have helped decrease the percentage of misspelled words.

4.5.2 Limiting Character length

In most cases, a Tweet's text content can contain up to 280 characters or Unicode glyphs [9], where some glyphs may count as more than one character. As a result, we have limited the character count to 280 for each row.

4.5.3 Data Augmentation by Replication

To augment the data, we randomly select 15 percent portion of rows from the original dataset and duplicate them. To improve the effectiveness of our learning, we give priority to tweets with longer word lengths. This not only adds more rows to the dataset but also enhances the quality of the data.

[9] Source:https://developer.twitter.com/en/docs/counting-characters#:˜text=In%20most%20cases%2C%20the%20text,as%20more%20than%20one%20character.

4.5.4 Class Balancing

We employed class balancing as a technique for handling class imbalance in the sentiment analysis task. The dataset that I used consisted of 3 class labels - positive, negative, and neutral - and each class had 19000 rows. However, the distribution of samples across the classes was not balanced, with some classes having significantly more samples than others.

To address this issue, I used the class balancing technique, which involves randomly oversampling the minority class and undersampling the majority class to achieve a balanced distribution of samples across the classes. In this case, I undersampled both the positive and negative classes to achieve a balanced distribution of samples across all three classes.

By employing class balancing, I was able to improve the performance of the sentiment analysis model by reducing the bias towards the majority class and ensuring that the model is equally sensitive to all classes.

Table 4.8: Sentiment Class Labels Balanced

Sentiment	Count
Negative	19000
Neutral	19000
Positive	19000

4.5.5 Data Spliting

When working with a large dataset, it is important to split the data into different subsets for training, validation, and testing. The convention for splitting the data is often to use a ratio of 60:20:20 for training, validation, and testing, respectively. However, depending on the size of the dataset, the exact ratio can vary. In our case, since we have a large combined dataset, we have decided to use a new ratio of 80:10:10 for training, validation, and testing, respectively. We took care to achieve class balancing while splitting the data, which is called stratifying.

4.5.6 DataLoader

Similar to TFRecords that was used in this study[18], PyTorch DataLoader allows for efficient storage and access to data in a binary file format. PyTorch DataLoader can handle various data formats, such as images, text, and audio, and can efficiently load and transform them into tensors that can be used as inputs to deep learning models. This can be particularly useful when dealing with large datasets that do not fit in memory, as it allows for efficient disk-based storage and retrieval of data.

4.5.7 Random Sampling

technique involves randomly selecting a subset of data points from the Train dataset, without any particular order or pattern, to form the training dataset. Random sampling can be used to create a diverse and representative training dataset, as it ensures that each data point has an equal chance of being included in the training dataset and that our model doesn't learn specific patterns from the ordering too [41].

4.6 Sentiment Classification

We conducted an experimentation on Sentiment Analysis, comparing the performance of Baseline Models with our competitive Ensemble model.

4.6.1 Baseline Models

Machine Learning Approaches [10], We conducted a comparative analysis of several machine learning models for multi-class sentiment classification. To select the models, we referred to previous studies in the field of sentiment analysis that reported exemplary performance. Additionally, we performed Randomized Search 5 times on each 5 Cross Validation (CV) on various hyperparameters, drawing insights from existing literature. The models tested in our study best hyperparameters, as depicted in the following figures :

Table 4.9: Decision Tree Model Hyperparameters

Hyperparameter	Value
TF-IDF ngram range	(1, 1)
TF-IDF max features	10,000
Classifier min samples split	5
Classifier min samples leaf	2
Classifier criterion	entropy

Table 4.10: Multinomial Naive Bayes Model Hyperparameters

Hyperparameter	Value
TF-IDF ngram range	(1, 1)
TF-IDF max features	20,000
Classifier alpha	1

[10] Notebook:https://colab.research.google.com/drive/18F75YeaO0hSJgZjciiit6-zIRABujTy?usp=sharing

Table 4.11: Random Forest Model Hyperparameters

Hyperparameter	Value
TF-IDF ngram range	(1, 2)
TF-IDF max features	20,000
Classifier min samples split	10
Classifier min samples leaf	4
Classifier n estimators	100

Table 4.12: Linear Support Vector Machine (SVM) Model Hyperparameters

Hyperparameter	Value
TF-IDF ngram range	(1, 1)
TF-IDF max features	10,000
Classifier loss	squared hinge
Classifier C	0.1

Table 4.13: Logistic Regression Model Hyperparameters

Hyperparameter	Value
TF-IDF ngram range	(1, 1)
TF-IDF max features	10,000
Classifier penalty	L2
Classifier C	1
Classifier max iter	600

Table 4.14: K-Nearest Neighbors (KNN) Model Hyperparameters

Hyperparameter	Value
TF-IDF ngram range	(1, 1)
TF-IDF max features	5,000
Classifier n neighbors	5

Table 4.15: AdaBoost Model Hyperparameters

Hyperparameter	Value
TF-IDF ngram range	(1, 2)
TF-IDF max features	10,000
Classifier n estimators	200
Classifier learning rate	0.5
Classifier base estimator	Linear SVM (C=0.1)
Classifier algorithm	SAMME

Table 4.16: XGBoost Model Hyperparameters

Hyperparameter	Value
TF-IDF ngram range	(1, 2)
TF-IDF max features	5,000
Classifier max depth	10
Classifier n estimators	50
Classifier learning rate	0.5

Deep Learning Approaches [11], We also conducted a comparative analysis of several deep learning models for multi-class sentiment classification. To select the model's architecture and hyperparameters as shown in this Figure 4.15, we referred to previous studies[6][26] in the field of sentiment analysis that reported exemplary performance. Drawing insights from existing literature. In addition, We added a new Text preprocessing step before the training which is Transliteration is the process of representing the Arabic script using the Latin alphabet. Here are some common transliterated words (سلام), Jazak Allah khair (جزاك الله خيراً).

Table 4.17: Recommended Optimal hyper-parameter values. Adapted from:[6]

Hyper-parameter	Values
Batch size	32
Learning rate (Adam)	0.0001
Number of epochs	25
Early stopping checkpoint	14

Batch Normalization the activations of each layer are normalized by subtracting the mean and dividing by the standard deviation of the activations within the mini-batch. This helps to reduce the internal covariate shift, which is the phenomenon where the distribution of the input to each layer changes as the parameters of the previous layers change during training. By normalizing the activations at each mini-batch, batch normalization can make the learning process more stable and help to prevent overfitting.

Layer Normalization is a type of normalization that is applied at the layer level, as opposed to instance normalization or batch normalization which are applied at the level of individual examples or mini-batches. The activations of each layer are normalized by subtracting the mean and dividing by the standard deviation of the activations within that layer. This helps to reduce the covariate shift, which is the phenomenon where the distribution of the input to each layer changes as the parameters of the previous layers

[11] DL Notebook: https://colab.research.google.com/drive/1Z2HouT1JXS27tO2sil8i$_t$9n18$siaD4Q?usp = sharing$

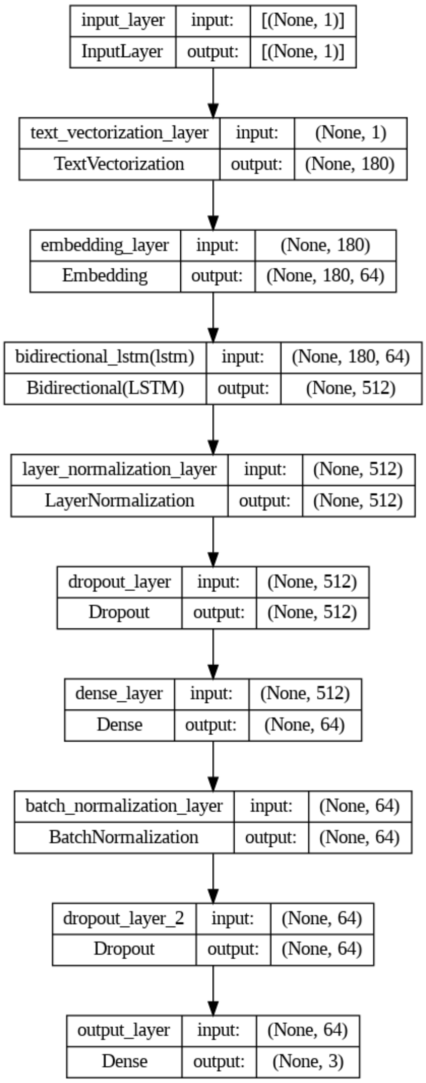

input_layer | input: | [(None, 1)]
InputLayer | output: | [(None, 1)]
text_vectorization_layer | input: | (None, 1)
TextVectorization | output: | (None, 180)
embedding_layer | input: | (None, 180)
Embedding | output: | (None, 180, 64)
bidirectional_lstm(lstm) | input: | (None, 180, 64)
Bidirectional(LSTM) | output: | (None, 512)
layer_normalization_layer | input: | (None, 512)
LayerNormalization | output: | (None, 512)
dropout_layer | input: | (None, 512)
Dropout | output: | (None, 512)
dense_layer | input: | (None, 512)
Dense | output: | (None, 64)
batch_normalization_layer | input: | (None, 64)
BatchNormalization | output: | (None, 64)
dropout_layer_2 | input: | (None, 64)
Dropout | output: | (None, 64)
output_layer | input: | (None, 64)
Dense | output: | (None, 3)

Figure 4.15: Model Architecture

change during training. By normalizing the activations at each layer, layer normalization can make the learning process more stable and help to prevent overfitting. It is an efficient technique because it is parallelizable[19].

4.6.2 Transformers Fine-tuning

We utilized the HuggingFace[12] Transformers library to fine-tune pre-trained language models, which offers a PyTorch interface. Specifically, we selected the CAMel-Mix, CAMel-da, AraBERT, MARBERT[13] pre-trained model [14]. To prepare the data, we

[12] HuggingFace : https://huggingface.co/

[13] MARBERT : https://huggingface.co/Ammar-alhaj-ali/arabic-MARBERT-sentiment

[14] Ensemble NoteBook: https://colab.research.google.com/drive/1nn91vLfiwN-JASkuUdzVnXB5-lSj3XIn?usp=sharing

needed to specify the maximum length of our sentences for padding and truncating. Therefore, after the first step of data preparation, we tokenized the dataset once to record the lengths of each tokenized tweet. We plotted the distribution of tokenized tweet lengths and discovered that the majority of tweets contained fewer than 50 tokens, as depicted in Figure 4.11. To make sure that there is an adequate margin or reserve, we set the maximum length to 80.

We employed on the 5 different Arabic BERT model the same fine-tuning approach outlined in section 3.3, which involves adding a feed-forward neural network classification layer on top of BERT, with a dropout rate of 0.3. Despite experimenting with various hidden layer sizes (50, 128, and 512), the results remained consistent across all sizes. The experiments were carried out manually, with multiple runs of the model using different numbers of layers, activation functions, hidden layer sizes, normalization techniques, dropout rates, and a comparison of the final results. The loss function employed in this study was Sparse Categorical Cross-Entropy (SCCE) with Focal Loss, with equal weights assigned to each class, as we ensured that our dataset was equally balanced.

While optimal hyper-parameter values are task-specific, the authors of several models have recommended certain hyper-parameters for fine-tuning that they found to be effective for most tasks. Table 4.18 displays these recommended values.

Table 4.18: Recommended hyper-parameter values. Adapted from:[37][18] [4]

Hyper-parameter	Values
Batch size	16 ,32 or 64
Learning rate (Adam)	1e-5 to 5e-5
Number of epochs	10
Early stopping checkpoint	4

We manually tested different combinations of these values. The following Tables shows the hyper-parameter combination that achieved the highest accuracy for each Model, according to our experiments.

Table 4.19: Optimal hyper-parameters for the sentiment analysis task for Arabert.

Hyper-parameter	Values
Batch size	32
Learning rate (Adam)	5e-5
Number of epochs	10
Early stopping checkpoint	8

Table 4.20: Optimal hyper-parameters for the sentiment analysis task for MARBERT.

Hyper-parameter	Values
Batch size	32
Learning rate (Adam)	3e-5
Number of epochs	10
Early stopping checkpoint	4

Table 4.21: Optimal hyper-parameters for the sentiment analysis task for CAMel-mix and CAMel-da.

Hyper-parameter	Values
Batch size	32
Learning rate (Adam)	5e-5
Number of epochs	10
Early stopping checkpoint	4

4.7 Sentiment Forecasting

I collected a dataset of English Tweets and translated them into Arabic using a translation model. The translated tweets were then preprocessed and tokenized using the CAMel-BERT-Mix tokenizer. I used our Arabic Ensemble Model to predict the sentiment of the tweets.

To train an LSTM model for sentiment forecasting, I used a sliding window of 10 tweets as input and the sentiment of the current tweet as output. The dataset was split into training, validation, and test sets, with an 80:10:10 ratio. The LSTM model had a 64-unit LSTM layer, a Dense layer with 8 hidden units using relu activation function, and a classification Dense layer with 3 hidden units using softmax activation function.

To select the model's hyperparameters as shown in this Figure4.22, we experimented with different hyperparameters, and we choose the loss function to be the root mean squared error.

Table 4.22: Optimal hyper-parameter values.

Hyper-parameter	Values
Batch size	64
Learning rate (Adam)	0.0001
Number of epochs	50

I evaluated the LSTM model's performance on the test set, which included tweets from different months, and the results showed that the model achieved high accuracy in sentiment forecasting. This approach could be useful in analyzing the sentiment of Arabic tweets over time and understanding trends and patterns in social media conversations.

4.8 Topic Modeling

We conducted a comparative analysis of the topics discussed in our Balanced Tweets Dataset without performing any splitting by using BERTopic, and compared its performance with the other 2 baseline models LDA and NMF. We also experimented with different settings for the BERTopic embedding model whether AraBERT or CAMel-MIX-Bert Embeddings and ultimately decided to use CAMel-MIX-Bert Embeddings as it achieves a higher Coherence Score 4.9.2, with the number of topics set to five.

Following this, we will produce the top five topics and their corresponding word clouds, and statically label them to assess their relevance and coherence with respect to our dataset.

In summary, our objective was to utilize BERTopic to gain a more comprehensive understanding of the topics discussed in the Balanced Tweets Dataset and to assess its performance against traditional baseline models.

4.9 Classification evaluation methods

Classification evaluation methods are used to evaluate the performance of a classification model by comparing the predicted labels with the actual labels of a dataset.

4.9.1 Cross-validation

Cross-validation is a widely used technique to evaluate the performance of a machine learning model in sentiment analysis. In cross-validation, the dataset is divided into k-folds, where k is typically 5 or 10. The model is trained on k-1 folds and tested on the remaining fold. This process is repeated k times, with each fold used exactly once for testing. The results of each fold are then averaged to obtain the overall performance of the model.[54]

4.9.2 Confusion Matrix

which is a table that summarizes the actual and predicted labels of a classification model. which are True Positive (TP), False Positive(FP), True Negative (TN), and False Negative(FN).

Accuracy

The proportion of correctly classified instances to the total number of instances in a dataset.

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN} \tag{4.1}$$

Precision

$$Precision = \frac{TP}{TP + FP} \tag{4.2}$$

The proportion of true positives to the total number of instances predicted as positive by the model. It measures how precise the model's positive predictions are.

Recall (Sensitivity or True Positive Rate):

$$Recall = \frac{TP}{TP + FN} \tag{4.3}$$

The proportion of true positives to the total number of actual positive instances in a dataset. It measures how well the model is able to identify positive instances.

F1 Score:

$$F1 = 2 * \frac{Precision * Recall}{Precision + Recall} \tag{4.4}$$

It is a combined metric that takes both precision and recall into account, Rijsbergen [22] defined it as the harmonic mean of precision and recall.

Coherence Score:

is a measure of the interpretability or coherence of a topic model. It measures the degree of semantic similarity between the most representative words of a given topic, based on their co-occurrence in the documents. The coherence score is often used as an evaluation metric to compare and optimize different topic models.

4.10 Results Analysis and Discussion

In this analysis, we will evaluate the performance and outcomes of Baseline Models (Machine Learning/Deep Learning) in Sentiment Analysis and compare them with our Ensemble model. By comparing the results, we aim to make informed decisions regarding the effectiveness of the Ensemble model in this task.

4.10.1 Sentiment Classification

The performance results of each model (whether ML/DL/Transformers) independently on the Randomly Sampled 10% of rows from the Balanced Merged Dataset that we prepared in Section 4.5, which is the Validation dataset are shown in the following Tables: Table 4.23 are results of the ML baseline models that are not considerably different when compared to one another. However, the Different Arabic BERT models exhibited a significant improvement.

Table 4.23: Sorted Machine Learning Text Classifier Results using TF-IDF with N-grams and stemming

Name	F1 (macro)	F1 (weighted)	Precision	Recall	Accuracy
LogisticRegression	**0.74**	**0.74**	**0.75**	**0.74**	**0.74**
LinearSVC	0.74	0.74	0.74	0.74	0.74
RandomForest	0.73	0.73	0.73	0.73	0.73
Multinomial NB	0.71	0.71	0.71	0.71	0.71
XGBOOST	0.70	0.70	0.71	0.70	0.70
Decesion Tree	0.65	0.65	0.65	0.65	0.65
KNeighborsClassifier	0.63	0.63	0.64	0.63	0.63
AdaBoost(LogisticRegression)	0.62	0.62	0.65	0.62	0.62

Table 4.24: Training and validation results for the model over 14 epochs.

Epoch	Training Loss	Training Accuracy	Validation Accuracy
0	2.3011	0.4743	0.5246
1	1.9941	0.5745	0.6117
2	1.8244	0.6299	0.6490
3	1.6875	0.6677	0.6769
4	1.5642	0.6992	0.6986
5	1.4532	0.7249	0.7069
6	1.3458	0.7505	0.7172
7	1.2500	0.7713	0.7237
8	1.1607	0.7914	0.7262
9	1.0739	0.8095	0.7255
10	0.9953	0.8252	0.7258
11	0.9239	0.8365	0.7254
12	0.8576	0.8497	0.7221
13	**0.8018**	**0.8581**	**0.7215**

Table 4.25: CAMel-DA Training and validation results

Epoch	Train loss	Val loss	Accuracy	F1 (weighted)	F1 (macro)	Precision	Recall
0	**0.584**	**0.535**	**0.791**	**0.791**	**0.784**	**0.807**	**0.791**
1	0.362	0.568	0.787	0.787	0.780	0.803	0.787
2	0.212	0.805	0.784	0.784	0.777	0.800	0.784
3	0.149	0.874	0.782	0.782	0.775	0.799	0.782

Table 4.26: CAMel-MIX Training and validation results

Epoch	Train Loss	Val. Loss	Accuracy	F1 (weighted)	F1 (macro)	Precision	Recall
0	0.582	0.547	0.785	0.785	0.777	0.805	0.785
1	**0.381**	**0.544**	**0.793**	**0.792**	**0.786**	**0.807**	**0.793**
2	0.231	0.719	0.790	0.789	0.782	0.804	0.790
3	0.163	0.810	0.786	0.786	0.779	0.802	0.786
4	0.127	1.049	0.783	0.782	0.775	0.798	0.783
5	0.100	1.033	0.779	0.779	0.772	0.796	0.779

Table 4.27: Alanzi Training and validation results

Epoch	Train Loss	Val Loss	Accuracy	F1 (weighted)	F1 (macro)	Precision	Recall
0	**0.584**	**0.535**	**0.791**	**0.791**	**0.784**	**0.807**	**0.791**
1	0.362	0.568	0.787	0.787	0.780	0.803	0.787
2	0.212	0.805	0.784	0.784	0.777	0.800	0.784
3	0.149	0.874	0.782	0.782	0.775	0.799	0.782

Table 4.28: MARBERT Training and validation results

Epoch	Train Loss	Val Loss	Accuracy	F1 (weighted)	F1 (macro)	Precision	Recall
0	0.571	0.515	0.799	0.800	0.791	0.822	0.799
1	**0.363**	**0.533**	**0.801**	**0.800**	**0.791**	**0.816**	**0.801**
2	0.222	0.692	0.799	0.799	0.790	0.813	0.799
3	0.156	0.847	0.795	0.795	0.787	0.810	0.795

Table 4.29: AraBERT Training and validation results

Epoch	Train Loss	Val Loss	Accuracy	F1 (weighted)	F1 (macro)	Precision	Recall
0	1.104	1.098	0.347	0.184	0.170	0.127	0.347
1	1.101	1.098	0.347	0.184	0.170	0.127	0.347
2	1.099	1.098	0.347	0.184	0.170	0.127	0.347
3	**1.099**	**1.097**	**0.347**	**0.184**	**0.170**	**0.127**	**0.347**

As we can see in Table 4.29 for AraBERT, underfitting occurs and there are no significant improvements during training. Despite attempts to change the classification layer and hyperparameters, underfitting persists, leading us to discard this model from our Ensemble Model.

Next, we tested the 4 other Arabic Models namely: MARBERT, CaMel-bert-DA, CaMel-bert-Mix, and Alanzi on the Test set and calculated their accuracies to use them later.

Table 4.30: CAMel-DA Test results

Loss	Accuracy	F1 (weighted)	F1 (macro)	Precision	Recall
0.535	0.791	0.792	0.784	0.808	0.791

Table 4.31: CAMel-MIX Test Results

Loss	Accuracy	F1 (weighted)	F1 (macro)	Precision	Recall
0.543	0.792	0.792	0.785	0.807	0.792

Table 4.32: Alanzi Test Results

Loss	Accuracy	F1 (weighted)	F1 (macro)	Precision	Recall
0.466	0.838	0.838	0.831	0.852	0.838

Table 4.33: MARBERT Test Results

Loss	Accuracy	F1 (weighted)	F1 (macro)	Precision	Recall
1.303	0.368	0.366	0.357	0.391	0.368

After calculating the accuracy of each model, we will normalize the accuracies to ensure they sum up to 1. This will be done by dividing each model's accuracy by the sum of the accuracies of all the models. The resulting values will represent the contribution percentage of each model to the ensemble model. As shown in this Figure 4.34.

$$Model's\ Contribution\ percentage_i = \frac{ModelAccuracy}{\sum\limits_{i=1}^{4} ModelAccuracy_i} \tag{4.5}$$

Table 4.34: Ensemble model contributions

Name	**Contribution Percentage**	**Accuracy**
CAMeL-Lab/bert-base-arabic-camelbert-mix	0.28	0.792
CAMeL-Lab/bert-base-arabic-camelbert-da	0.28	0.791
Ammar-alhaj-ali/arabic-MARBERT-sentiment	0.13	0.369
ALANZI/imamu_arabic_sentimentAnalysis	0.3	0.8

We included MARBERT in our Ensemble model and allowed it to contribute to the predictions, as shown in Table 4.34. Despite having an accuracy below 0.5, we made this choice because we wanted to improve the model's ability to generalize to new data in the future. This decision reflects a potential trade-off between accuracy and generalization.

The key advantage of using the Transformer's accuracy based on the test set is that it allows our ensemble model to dynamically adjust to unseen data and Arabic dialects. Each Transformer model possesses the ability to make more accurate predictions for specific Arabic dialects, thus enabling our ensemble model to be adaptable. This adaptability can be achieved by testing and recreating the contribution percentage table, similar to the example shown in Table 4.34.

We normalize the prediction logits of each model by applying the Tanh function to them, which ensures that their values fall within the range of -1 to 1, accounting for negative and positive logits.

$$Normalized\ Logits_i = tanh(Model\ Prediction\ Logits_i) \tag{4.6}$$

We multiply each Normalized Logits 4.6 with their respective Model's Contribution percentage 4.5 and this will results :

$$New\ Normalized\ Logits_i = Model\ Prediction\ Logits_i Model's\ Contribution\ percentage_i \tag{4.7}$$

We add up all the new normalized logits of our model, as shown in Equation 4.7, and then apply the Softmax function to obtain a probability distribution for each sentiment. Finally, we choose the sentiment with the highest probability.

$$Positive_{prop}\ Negative_{prop_i}\ Neutral_{prop_i} = Softmax(\sum_{i=1}^{4} New\ Normalized\ Logits_i) \tag{4.8}$$

We evaluated our Ensemble Model on the Test Set and this is the results as depicted in this Figure 4.35 :

Table 4.35: Classification Report - Ensemble Model

	precision	recall	f1-score	support
positive	0.84	0.84	0.84	3500
negative	0.83	0.82	0.83	3712
neutral	0.81	0.82	0.82	3841
accuracy	0.83			
macro avg	0.83	0.83	0.83	11053
weighted avg	0.83	0.83	0.83	11053

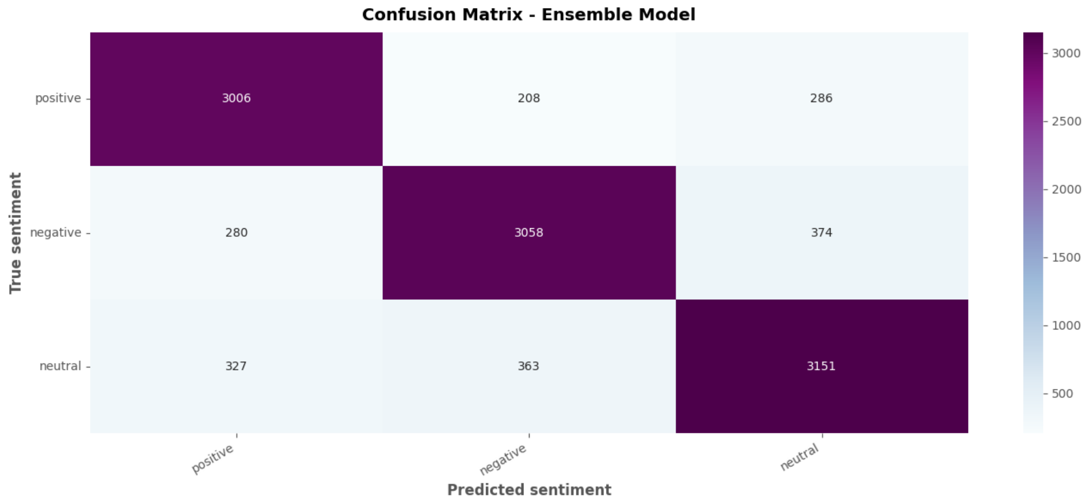


Figure 4.16: Confusion Matrix - Ensemble Model

The results depicted in both figures 4.16 and 4.35 clearly demonstrate the superior performance and accuracy of our ensemble model. Comparatively, it outperforms the standalone Transformer model by approximately 5% in prediction accuracy.

4.10.2 Sentiment Forecasting

The dataset used in this study was initially collected and subsequently translated using a translation model. After translation, the dataset underwent preprocessing steps to ensure data quality and consistency. The final dataset comprises a total of 223 samples.

Figure 4.17 presents an overview of the dataset's temporal distribution, showcasing the frequency of data points over time. Additionally, Figure 4.19 illustrates the distribution of sentiment labels within the dataset across different time periods.

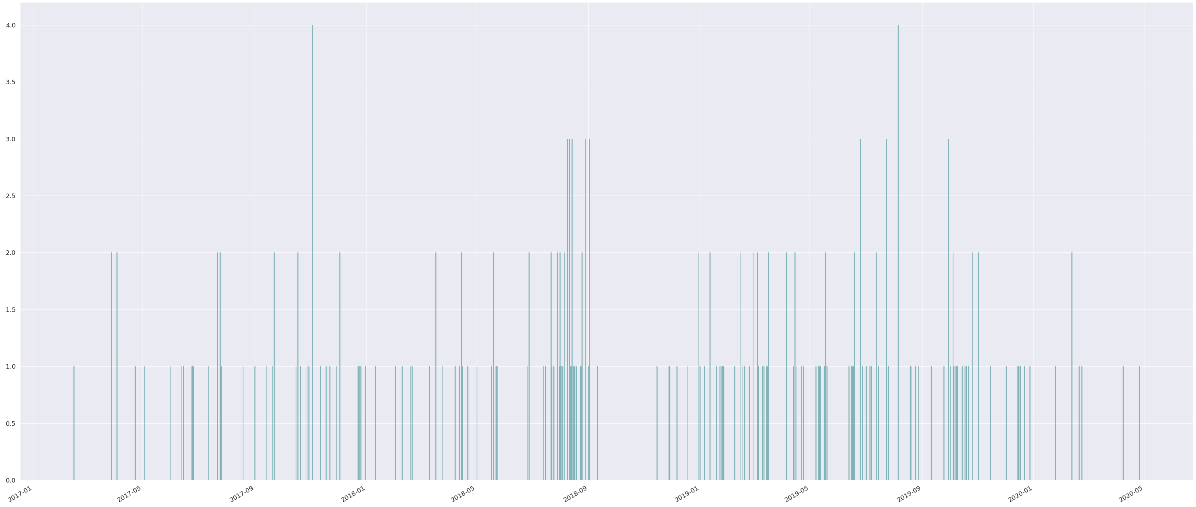


Figure 4.17: Tweets Timeline

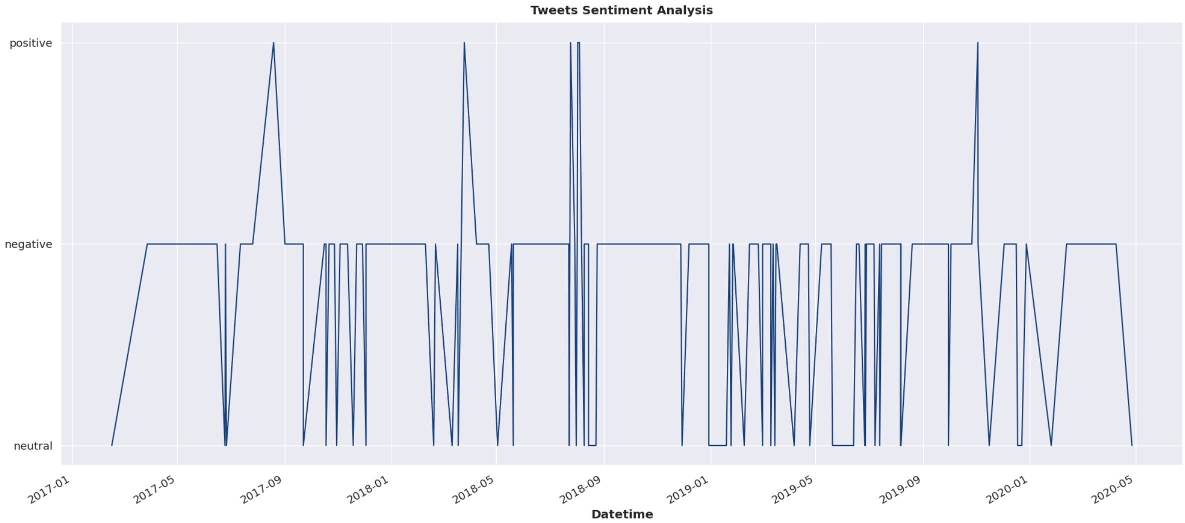


Figure 4.18: Original Predicted Sentiment Timeline - Whole DataSet

After training our model for 50 epochs, we conducted evaluations on multiple datasets:

the in-sample Train dataset, as well as the out-of-sample Validation and Test datasets. The evaluation results are presented in Figures 4.19, 4.20, and 4.21.

These figures provide visual representations of the confusion matrices, which showcase the performance of our model in predicting sentiment labels across different datasets. The confusion matrices offer valuable insights into the model's accuracy, precision, recall, and overall classification performance.

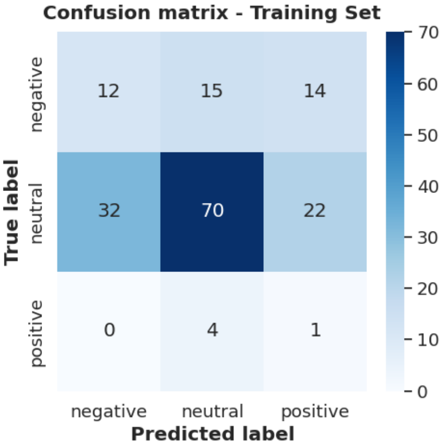


Figure 4.19: Confusion Matrix - Train Set Forecasting

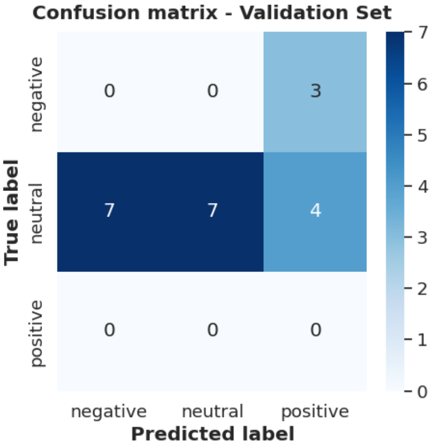


Figure 4.20: Confusion Matrix - Validation Set Forecasting

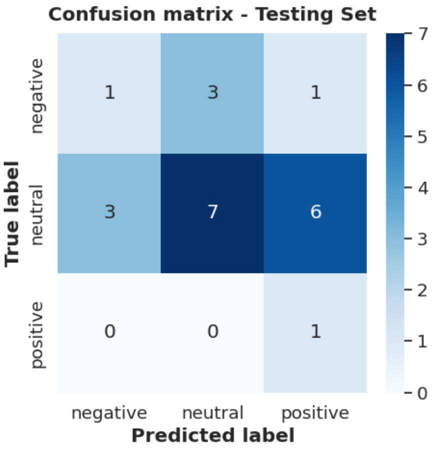


Figure 4.21: Confusion Matrix - Test Set Forecasting

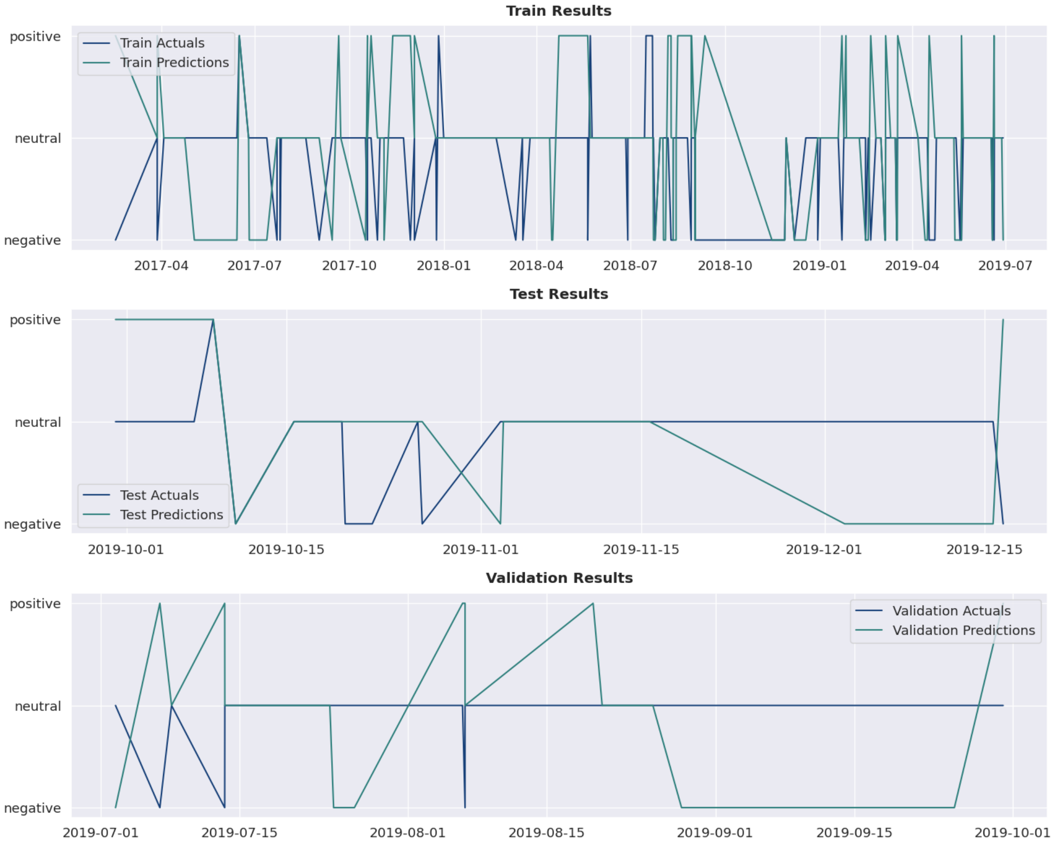


Figure 4.22: Timeline - Result Forecasting

Figure 4.22 provides a graphical representation that allows for an assessment of the model's performance in capturing the sentiment trends over time. By comparing the original distribution with the predicted distribution, we can gain insights into the accuracy and effectiveness of the model in forecasting sentiment dynamics.

Due to the scarcity of time-lined data and the limited number of patterns in sentiment distribution over time, our model encountered challenges in learning effectively. As a result, its accuracy was observed to be 49

The scarcity of time-lined data refers to the insufficient availability of data points with precise timestamps, which is crucial for capturing temporal patterns and trends accurately. Additionally, the limited number of patterns in sentiment distribution across different time periods restricted the model's ability to generalize and make accurate predictions.

The observed accuracy of 49% indicates that the model's performance is close to random guessing, highlighting the need for additional data or alternative approaches to improve sentiment forecasting in such conditions.

It is worth noting that this outcome underscores the importance of having a substantial and diverse dataset, particularly when dealing with time-dependent phenomena like sentiment analysis.

4.10.3 Topic Modeling

We trained Bertopic and the other Baseline Models[15] on the whole dataset :

Bertopic

Bertopic achieved a coherence score of -0.022 with Arabert Embeddings and -0.020 with CAMel-Mix Embeddings which indicates that the topics generated by the model are slightly less coherent than random. A negative coherence score suggests that the topics have a high degree of overlap and do not represent distinct concepts. In other words, the topics generated by the model are not very interpretable or meaningful

[15] Notebook:https://colab.research.google.com/drive/1SJKG2zJagVmVA9IQwGFBiZqsxlbXT29F?usp=sharing

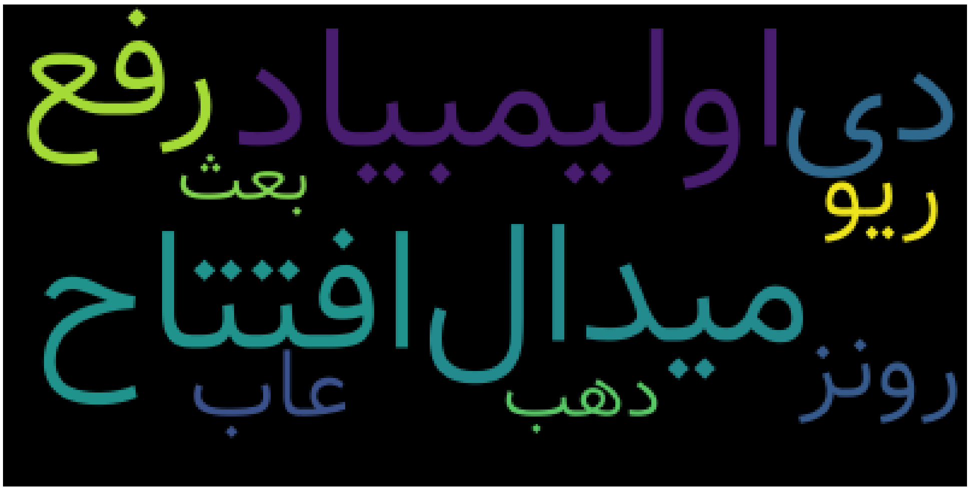


Figure 4.23: BERTopic - Topic 1 Word Cloud

This WordCloud (refer to Figure 4.23) appears to represent the topic of Sport Activity and Openings. The documents belonging to this topic have the following sentiment probabilities:

Table 4.36: Sentiment probabilities for Sport Activity and Openings topic

Sentiment	Probability
Positive	0.174
Neutral	0.619
Negative	0.206

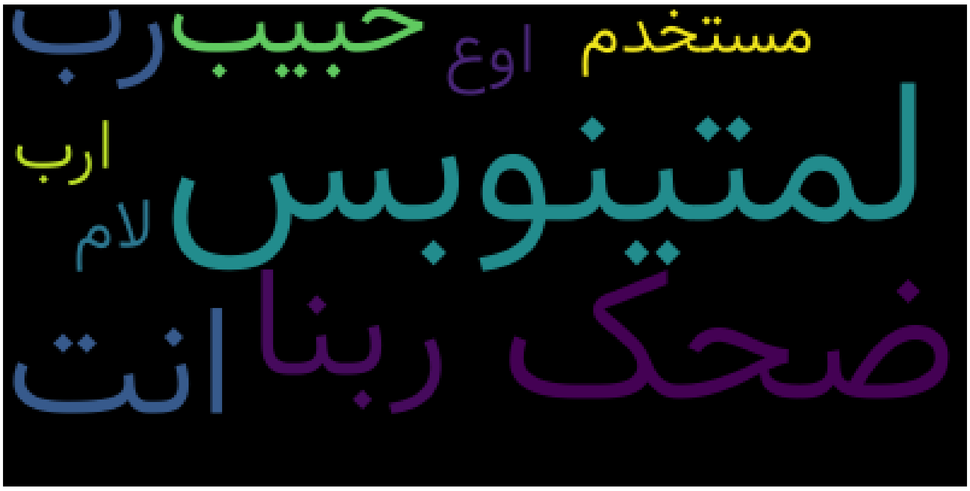


Figure 4.24: BERTopic - Topic 2 Word Cloud

This WordCloud (refer to Figure 4.24) appears to represent the topic of Religion and God. The documents belonging to this topic have the following sentiment probabilities:

Table 4.37: Sentiment probabilities for Religion and God topic

Sentiment	Probability
Positive	0.5
Neutral	0.205
Negative	0.294

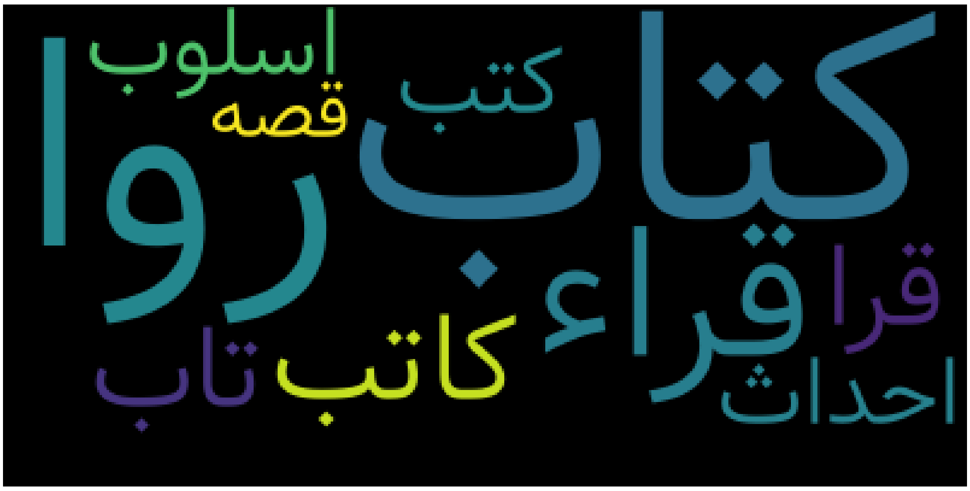


Figure 4.25: BERTopic - Topic 3 Word Cloud

This WordCloud (refer to Figure 4.25) appears to represent the topic of Book, Author, and Story Events. The documents belonging to this topic have the following sentiment probabilities:

Table 4.38: Sentiment probabilities for Book, Author, and Story Events topic

Sentiment	Probability
Positive	0.800
Neutral	0.015
Negative	0.183

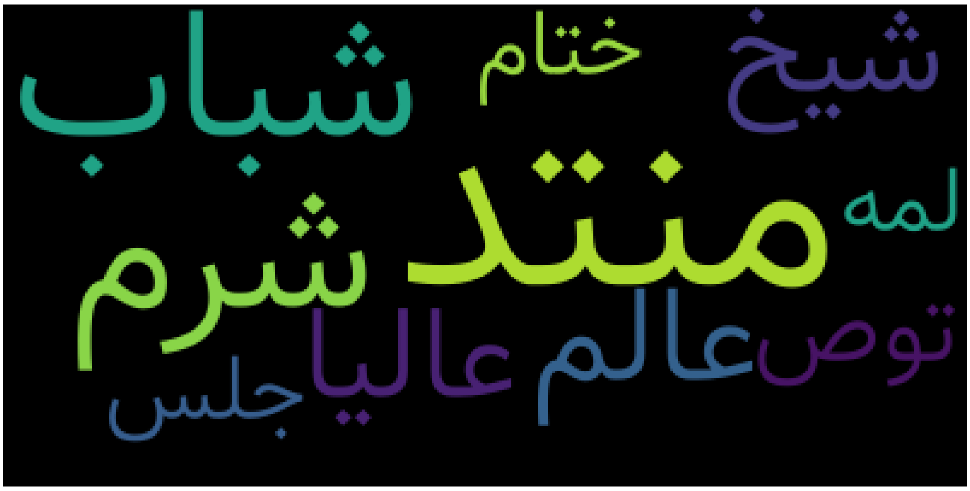


Figure 4.26: BERTopic - Topic 4 Word Cloud

This WordCloud (refer to Figure 4.26) appears to represent the topic of the Youth Forum in Sharm El-Sheikh. The documents belonging to this topic have the following sentiment probabilities:

Table 4.39: Sentiment probabilities for the Youth Forum in Sharm El-Sheikh topic

Sentiment	Probability
Positive	0.266
Neutral	0.676
Negative	0.05

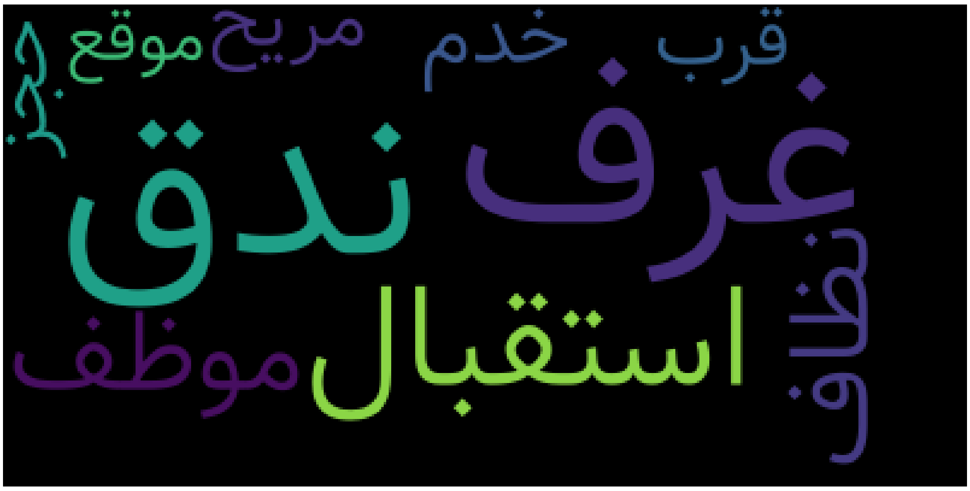


Figure 4.27: BERTopic - Topic 5 Word Cloud

This WordCloud (refer to Figure 4.27) appears to represent the topic of Hotel and

Room Reviews. The documents belonging to this topic have the following sentiment probabilities:

Table 4.40: Sentiment probabilities for the Hotel and Room Reviews topic

Sentiment	Probability
Positive	0.748
Neutral	0.025
Negative	0.226

Latent Dirichlet Allocation (LDA)

The LDA-based topic model scored a coherence score of -0.12623878278813425 so the model is not highly coherent but surprisingly it is better than Bertopic.

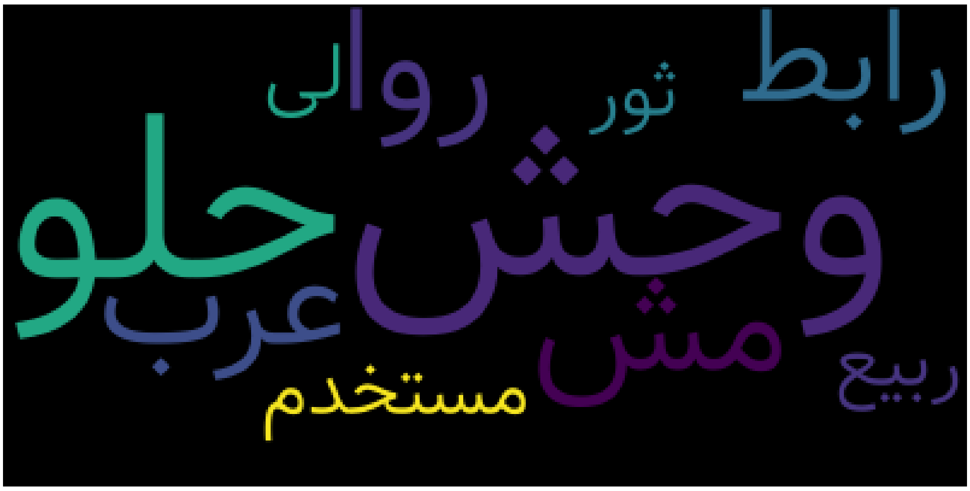


Figure 4.28: LDA - Topic 1 Word Cloud

This WordCloud (refer to Figure 4.28) appears to represent the topic of arab users. The documents belonging to this topic have the following sentiment probabilities:

Table 4.41: Sentiment probabilities for Arab users topic

Sentiment	Probability
Positive	0.255
Neutral	0.0252
Negative	0.491

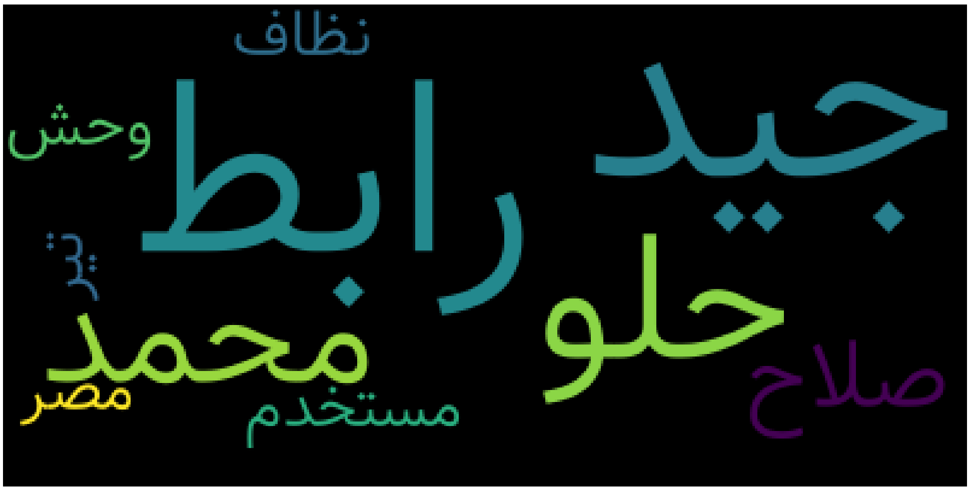


Figure 4.29: LDA - Topic 2 Word Cloud

This WordCloud (refer to Figure 4.29) appears to represent the topic of Mohamed Salah. The documents belonging to this topic have the following sentiment probabilities:

Table 4.42: Sentiment probabilities for Mohamed Salah topic

Sentiment	Probability
Positive	0.363
Neutral	0.34
Negative	0.295

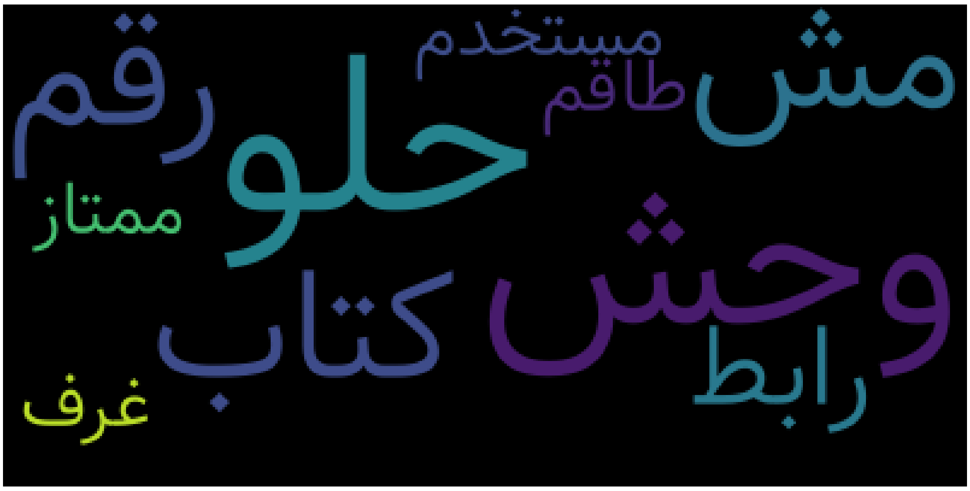


Figure 4.30: LDA - Topic 3 Word Cloud

This WordCloud (refer to Figure 4.30) appears to represent the topic of Hotel Room and Book Reviews. The documents belonging to this topic have the following sentiment probabilities:

Table 4.43: Sentiment probabilities for the Hotel Room and Book Reviews topic

Sentiment	Probability
Positive	0.267
Neutral	0.435
Negative	0.29

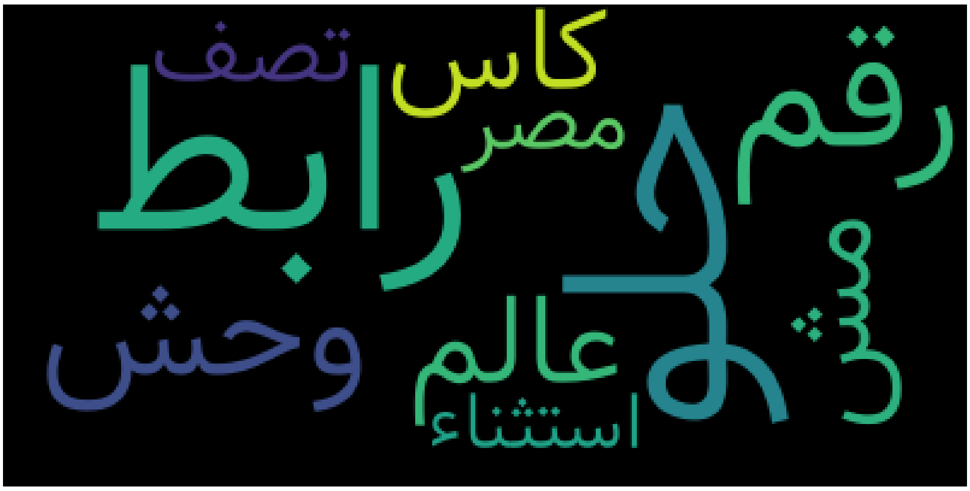


Figure 4.31: LDA - Topic 4 Word Cloud

This WordCloud (refer to Figure 4.31) appears to represent the topic of the World Cup in Egypt. The documents belonging to this topic have the following sentiment probabilities:

Table 4.44: Sentiment probabilities for the World Cup in Egypt topic

Sentiment	Probability
Positive	0.277
Neutral	0.177
Negative	0.544

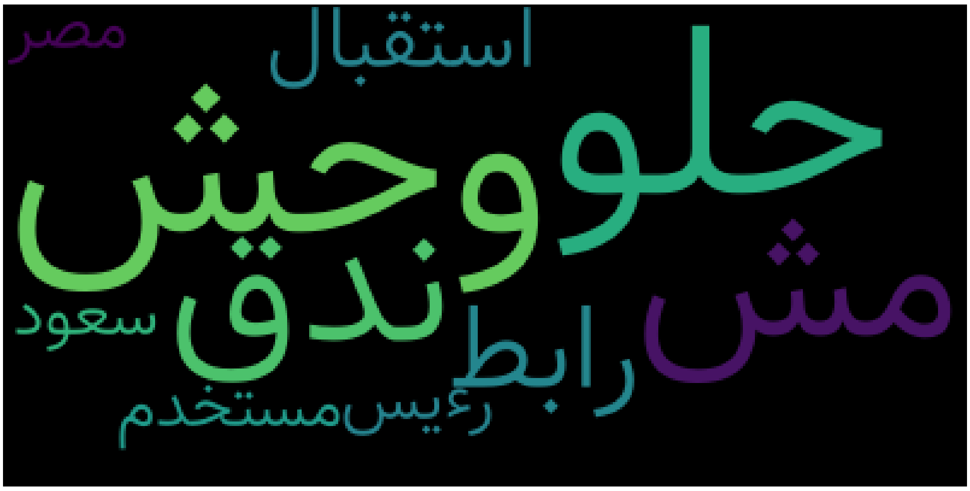


Figure 4.32: LDA - Topic 5 Word Cloud

This WordCloud (refer to Figure 4.32) appears to represent the topic of the President and the Army. The documents belonging to this topic have the following sentiment probabilities:

Table 4.45: Sentiment probabilities for the President and the Army topic

Sentiment	Probability
Positive	0.429
Neutral	0.272
Negative	0.298

Non-negative Matrix Factorization (NMF)

Surprisingly, The NMF-based topic model scored a coherence score of 0.15136070347068736 in which the generated topics are more coherent and interpretable than the model with a negative coherence score. which is best till now.

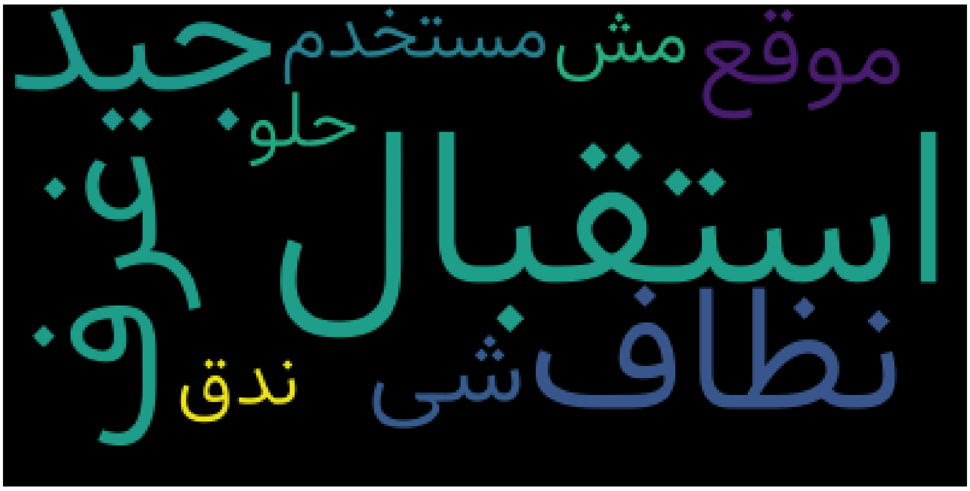


Figure 4.33: NMF - Topic 1 Word Cloud

This WordCloud (refer to Figure 4.33) appears to represent the topic of Hotel and Room Reviews. The documents belonging to this topic have the following sentiment probabilities:

Table 4.46: Sentiment probabilities for the Hotel and Room Reviews topic

Sentiment	Probability
Positive	0.602
Neutral	0.183
Negative	0.213

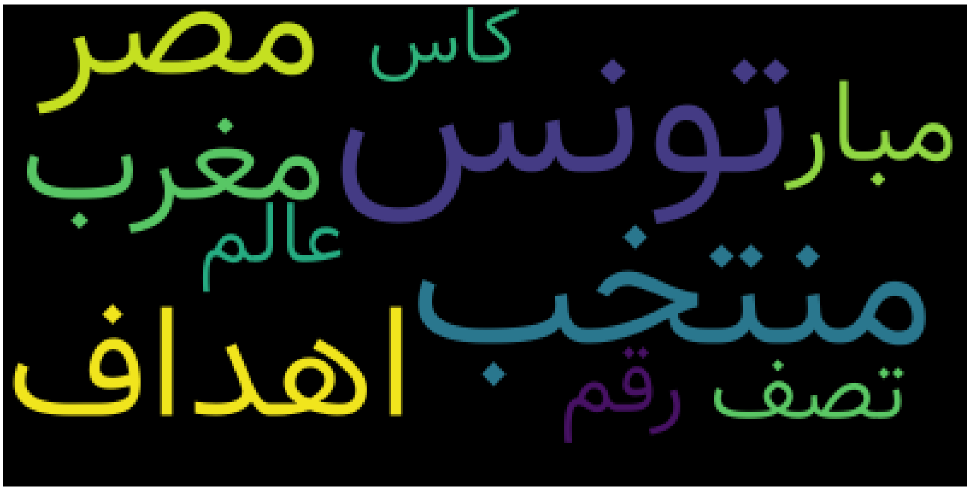


Figure 4.34: NMF - Topic 2 Word Cloud

This WordCloud (refer to Figure 4.34) appears to represent the topic of the Football World Cup. The documents belonging to this topic have the following sentiment probabilities:

Table 4.47: Sentiment probabilities for the Football World Cup topic

Sentiment	Probability
Positive	0.164
Neutral	0.700
Negative	0.135

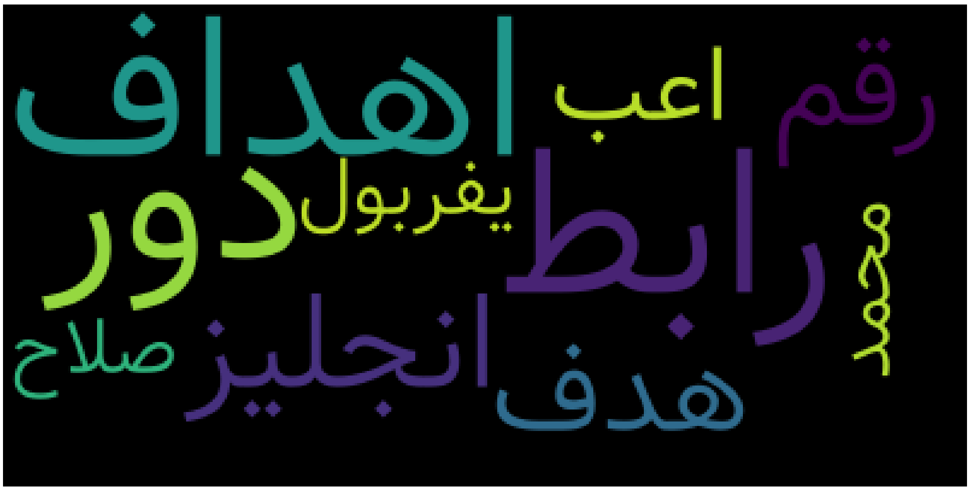


Figure 4.35: NMF - Topic 3 Word Cloud

This WordCloud (refer to Figure 4.35) appears to represent the topic of Liverpool and Mohamed Salah's Goals. The documents belonging to this topic have the following sentiment probabilities:

Table 4.48: Sentiment probabilities for Liverpool and Mohamed Salah's Goals topic

Sentiment	Probability
Positive	0.546
Neutral	0.333
Negative	0.120

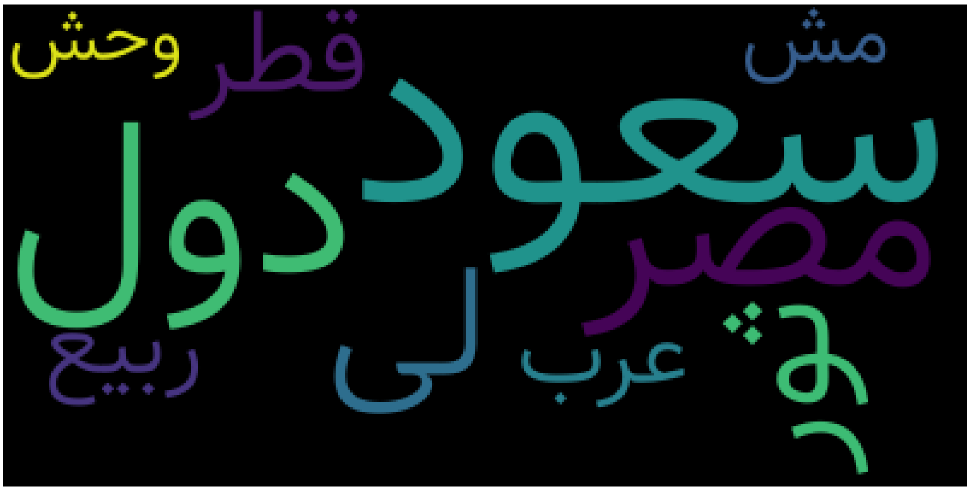


Figure 4.36: NMF - Topic 4 Word Cloud

This WordCloud (refer to Figure 4.36) appears to represent the topic of Arab Countries. The documents belonging to this topic have the following sentiment probabilities:

Table 4.49: Sentiment probabilities for the Arab Countries topic

Sentiment	Probability
Positive	0.158
Neutral	0.270
Negative	0.571

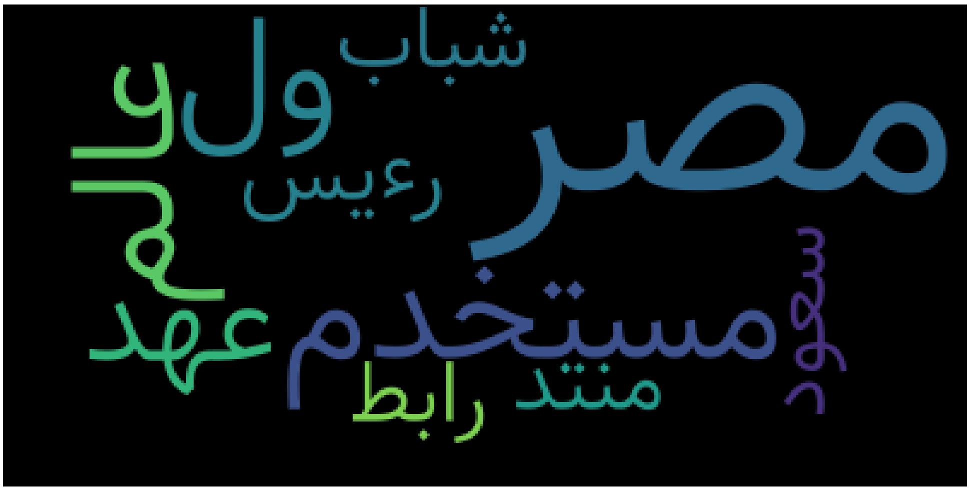


Figure 4.37: NMF - Topic 5 Word Cloud

This WordCloud (refer to Figure 4.37) appears to represent the topic of the President and Teenage. The documents belonging to this topic have the following sentiment probabilities:

Table 4.50: Sentiment probabilities for President and Teenage topic

Sentiment	Probability
Positive	0.223
Neutral	0.212
Negative	0.563

In comparing the word clouds generated by Bertopic, LDA (Latent Dirichlet Allocation), and NMF (Non-Negative Matrix Factorization), several key aspects can be considered.

Topic Representation and Sentiment Grouping:

- Bertopic: Bertopic utilizes the contextual embeddings of BERT (Bidirectional Encoder Representations from Transformers) to capture the semantic meaning of words. This enabled it to model the topics more accurately, fine-grained topic representation, and group similar words and sentiments in the same word cloud, and we can observe that each word cloud had a high probability distribution towards one sentiment only.

- LDA: LDA is a probabilistic topic modeling technique that represents topics as probability distributions over words. It assumes that each document is a mixture of topics and assigns probabilities to words within topics that's why we saw Good and Bad Semantic word repetition in every LDA word cloud because the dataset was mostly too about opinion tweets, and we can observe that LDA wasn't able to group documents that where sentimentally relevant.

- NMF: NMF is a matrix factorization technique that represents topics as linear combinations of non-negative basis vectors. It seeks to decompose the term-document matrix into a low-rank approximation, with each basis vector corresponding to a topic. This enabled the NMF model to capture underlying patterns and themes within the dataset, resulting in a more accurate and meaningful representation of topics, and we can observe that each word cloud had a high probability distribution towards one sentiment only.

Chapter 5

Conclusion & Future Work

5.1 Conclusion

This thesis aimed to investigate and analyze the sentiments expressed by individuals in their social media posts regarding a significant event. The objective was to track the evolution of sentiments over time by constructing a sentiment arc. To accomplish this, an ensemble architecture was designed, comprising four transformers: MARBERT, CaMel-bert-DA, CaMel-bert-Mix, and Alanzi. This ensemble architecture was followed by a final classification layer, consisting of a three-layer feed-forward neural network. The the output of each transformer applied on its logits a formula, and the logits were then summed before applying the softmax activation function.

The model was specifically fine-tuned for document-level sentiment analysis. Evaluating the model's performance on the sentiment analysis test dataset, an impressive test accuracy of 83% was achieved. This outcome underscores the significant contribution of Arabic BERT's language modeling capabilities in obtaining favorable results for sentiment analysis tasks, particularly for the Egyptian dialect.

Additionally, a sentiment forecasting deep learning model was developed, incorporating an LSTM architecture. This model demonstrated its effectiveness in accurately capturing and predicting sentiment patterns. However, due to the initially limited dataset availability, the achieved test accuracy was 49

Subsequently, during the exploration of topic modeling, BERTopic was employed with various pre-trained Arabic language models as embeddings. The outcomes of topic modeling and aspect-based analysis achieved by BERTopic were compared with those of LDA and NMF techniques. Notably, BERTopic and NMF exhibited comparable and competitive results, indicating their effectiveness in capturing meaningful topics. In contrast, LDA performed poorly in generating coherent and informative topics.

5.2 Future Work

Sentiment Model

The proposed model's effectiveness can be further investigated by experimenting with deeper layers and diverse architectures of the Classification Layer on various Arabic benchmark datasets and different Arabic Dialects. Moreover, it can be extended to address other Arabic sentiment analysis tasks, including aspect-based sentiment analysis.

Topic Modeling

While BERTopic showed promising initial results and demonstrated adaptability to different topic sizes, accurately evaluating the quality of the generated topics remains a challenging task that requires further research.

Sentiment Forecasting Model

In the context of sentiment forecasting, it is important to acknowledge that while the translation model from Hugging Face can be a valuable tool, it is not flawless, and certain nuances and complexities of the Arabic language may be lost in translation. It is essential to be aware of this limitation when utilizing translated data for sentiment analysis.

Furthermore, the choice of subword tokenization and the size of the sliding window are critical factors that can impact the performance of the sentiment forecasting model. Different preprocessing techniques and hyperparameters should be experimented with to optimize the model's performance and ensure accurate predictions.

The scarcity of data directly affects the model's ability to learn and capture the intricate patterns and dynamics of sentiment over time. When data is limited, models struggle to generalize and provide accurate forecasts, resulting in reduced accuracy levels.

To address this challenge and enhance sentiment forecasting, future research should prioritize efforts to expand the available dataset. This can be accomplished through various approaches, such as intensifying data collection efforts, exploring alternative data sources, and implementing data augmentation techniques. Augmenting the dataset will not only increase the volume of training examples but also provide a more comprehensive representation of sentiment across diverse domains, topics, and time periods, thereby improving the model's ability to capture nuanced patterns.

By recognizing the limitations of the translation model, optimizing preprocessing techniques, and addressing data scarcity through dataset expansion, future research can advance sentiment forecasting and enable more accurate predictions in real-world applications.

Appendix

Appendix A

Lists

List of Figures

Bibliography

[1] et al. A. Barhoumi. "an empirical evaluation of arabic-specific embeddings for sentiment analysis," in international conference on arabic language processing. *Springer. pp. 34-48*, 2019.

[2] Muhammad Abbas, K Ali Memon, A Aleem Jamali, Saleemullah Memon, and Anees Ahmed. Multinomial naive bayes classification model for sentiment analysis. *IJCSNS Int. J. Comput. Sci. Netw. Secur*, 19(3):62, 2019.

[3] Darwish K. Durrani N. Abdelali, A. and H. Mubarak. A fast and furious segmenter for arabic. in proceedings of the 2016 conference of the north american chapter of the association for computational linguistics: Demonstrations. 2016.

[4] Muhammad Abdul-Mageed, AbdelRahim Elmadany, and El Moatez Billah Nagoudi. Arbert & marbert: deep bidirectional transformers for arabic. *arXiv preprint arXiv:2101.01785*, 2020.

[5] Matthew England3 Abdulaziz M. Alayba1, Vasile Palade and Rahat Iqbal. Improving sentiment analysis in arabic using word representation. 2019.

[6] Ibrahim Abu Farha and Walid Magdy. Mazajak: An online Arabic sentiment analyser. In *Proceedings of the Fourth Arabic Natural Language Processing Workshop*, pages 192–198, Florence, Italy, August 2019. Association for Computational Linguistics.

[7] Abeer Abuzayed and Hend Al-Khalifa. Bert for arabic topic modeling: An experimental study on bertopic technique. *Procedia computer science*, 189:191–194, 2021.

[8] S. Al-Azani and E.-S. M. El-Alfy. "hybrid deep learning for sentiment polarity determination of arabic microblogs," in international conference on neural information processing. *Springer. pp. 491-500*, 2017.

[9] Mohammad Al-Smadi, Bashar Talafha, Mahmoud Al-Ayyoub, and Yaser Jararweh. Using long short-term memory deep neural networks for aspect-based sentiment analysis of arabic reviews. *International Journal of Machine Learning and Cybernetics*, 10:2163–2175, 2019.

[10] Ali Shatnawi Al-Tamimi, Abdel-Karim and Esraa Bani-Issa. Arabic sentiment analysis of youtube comments." 2017 ieee jordan conference on applied electrical engineering and computing technologies (aeect). 2017.

[11] A. Albayati and A. Al-Araji. Arabic sentiment analysis (asa) using deep learning approach,. 2020.

[12] Omar. Alharbi. A deep learning approach combining cnn and bi-lstm with svm classifier for arabic sentiment analysis." international journal of advanced computer science and applications 12.6. 2021.

[13] Mohamed Alkaoud and Mairaj Syed. On the importance of tokenization in arabic embedding models. In *Proceedings of the fifth Arabic natural language processing workshop*, pages 119–129, 2020.

[14] Abdelwahab A. Abdelkader H. Alqasemi. F. constructing automatic domain-specific sentiment lexicon using knn search via terms discrimination vectors. international journal of computers and applications, 41(2)129-139. 2019.

[15] A. A. Altowayan and L. Tao. "word embeddings for arabic sentiment analysis," in 2016 ieee international conference on big data (big data). *IEEE. pp. 3820-3825*, 2016.

[16] Ahmed Rafea Amira Shoukry. Preprocessing egyptian dialect tweets for sentiment mining. 2019.

[17] Olga Kovaleva Anna Rogers and Anna Rumshisky. A primer in bertology: What we know about how bert works. 2018.

[18] Wissam Antoun, Fady Baly, and Hazem Hajj. Arabert: Transformer-based model for arabic language understanding. *arXiv preprint arXiv:2003.00104*, 2020.

[19] Niki Parmar Jakob Uszkoreit Llion Jones Aidan N. Gomez Lukasz Kaiser Ashish Vaswani, Noam Shazeer and Illia Polosukhin. Attention is all you need. 1998.

[20] Bhavika Bhutani, Neha Rastogi, Priyanshu Sehgal, and Archana Purwar. Fake news detection using sentiment analysis. In *2019 twelfth international conference on contemporary computing (IC3)*, pages 1–5. IEEE, 2019.

[21] M. Biniz. "dataset for arabic classification,". 2018.

[22] David C. Blair. Information retrieval, 2nd ed. c.j. van rijsbergen. london: Butterworths. 1979.

[23] Naaima Boudad, Rdouan Faizi, Rachid Oulad Haj Thami, and Raddouane Chiheb. Sentiment analysis in arabic: A review of the literature. *Ain Shams Engineering Journal*, 9(4):2479–2490, 2018.

[24] Naaima Boudad, Rdouan Faizi, Rachid Oulad Haj Thami, and Raddouane Chiheb. Sentiment analysis in arabic: A review of the literature. *Ain Shams Engineering Journal*, 9(4):2479–2490, 2018.

[25] Kareem Darwish, Walid Magdy, et al. Arabic information retrieval. *Foundations and Trends® in Information Retrieval*, 7(4):239–342, 2014.

[26] Hanane Elfaik and El Habib Nfaoui. Deep bidirectional lstm network learning-based sentiment analysis for arabic text. *Journal of Intelligent Systems*, 30(1):395–412, 2020.

[27] Ashkan Eliasy and Justyna Przychodzen. The role of ai in capital structure to enhance corporate funding strategies,. 2020.

[28] Maryam Ayman ElOraby. Mapping and tracking sentiment arcs in social media streams. 2022.

[29] Steven Elsworth and Stefan Güttel. Time series forecasting using lstm networks: A symbolic approach. *arXiv preprint arXiv:2003.05672*, 2020.

[30] Christiane Fellbaum. Wordnet: An electronic lexical database. bradford books. 1998.

[31] Alec Go, Richa Bhayani, and Lei Huang. Twitter sentiment classification using distant supervision. *Processing*, 150, 01 2009.

[32] AEDA Hamouda and Fatma El-zahraa El-taher. Sentiment analyzer for arabic comments system. *Int. J. Adv. Comput. Sci. Appl*, 4(3), 2013.

[33] Abe Kazemzadeh François Bar Hao Wang, Dogan Can and Shrikanth Narayanan. A system for real-time twitter sentiment analysis of 2012 u.s. presidential election cycle. in proceedings of the acl 2012 system demonstrations, pages 115–120, jeju island, korea, july 2012. association for computational linguistics. 2020.

[34] Hansika Hewamalage, Christoph Bergmeir, and Kasun Bandara. Recurrent neural networks for time series forecasting: Current status and future directions. *International Journal of Forecasting*, 37(1):388–427, 2021.

[35] C. Hutto and Eric Gilbert. Vader: A parsimonious rule-based model for sentiment analysis of social media text. proceedings of the international aaai conference on web and social media. 2014.

[36] Mohammed V University Rabat Morocco Ikram El Karfi, Sanaa El Fkihi ENSIAS. An ensemble of arabic transformer-based models for arabic sentiment analysis. 2022.

[37] Go Inoue, Bashar Alhafni, Nurpeiis Baimukan, Houda Bouamor, and Nizar Habash. The interplay of variant, size, and task type in arabic pre-trained language models. *arXiv preprint arXiv:2103.06678*, 2021.

[38] Kenton Lee Jacob Devlin, Ming-Wei Chang and Kristina Toutanova. Bert: pre-training of deep bidirectional transformers for language understanding. 2018.

[39] Motaz Saad Kathrein Abu Kwaik. An arabic tweets sentiment analysis dataset (atsad) using distant,supervision and self training. 2020.

[40] Mohamed Maamouri, Ann Bies, Seth Kulick, Michael Ciul, Nizar Habash, and Ramy Eskander. Developing an egyptian arabic treebank: Impact of dialectal morphology on annotation and tool development. In *LREC*, pages 2348–2354, 2014.

[41] Xiangrui Meng. "scalable simple random sampling and stratified sampling."international conference on machine learning. pmlr. 2013.

[42] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[43] et al. Mukhoti, Jishnu. "calibrating deep neural networks using focal loss." advances in neural information processing system. *15288-15299*, 2020.

[44] D. Alsaeed H. Al-Baity R. Alshalan, H. Al-Khalifa and S. Alshalan. "detection of hate speech in covid-19–related tweets in the arab region: Deep learning and topic modeling approach," j. med. internet res. 2020.

[45] H. Z. R. M. Alahmary, Al-Dossari and A. Z. Emam. "sentiment analysis of saudi dialect using deep learning techniques," in 2019 international conference on electronics, information, and communication (iceic). *IEEE. pp. 1-6, 2019.*, 2019.

[46] Alec Radford and Karthik Narasimhan. Improving language understanding by generative pre-training. 2018.

[47] A. Rafea and N. GabAllah. A. rafea and n. gaballah, "topic detection approaches in identifying topics and events from arabic corpora," procedia comput. sci. *Procedia computer science*, 142:270–277, 2018.

[48] Ghulam Musa Raza, Zainab Saeed Butt, Seemab Latif, and Abdul Wahid. Sentiment analysis on covid tweets: an experimental analysis on the impact of count vectorizer and tf-idf on sentiment predictions using deep learning models. In *2021 International Conference on Digital Futures and Transformative Technologies (ICoDT2)*, pages 1–6. IEEE, 2021.

[49] Anoop S. and Asharaf S. Aspect-oriented sentiment analysis: A topic modeling-powered approach. *Journal of Intelligent Systems*, 29, 12 2018.

[50] Zakky Nilem Sanjifa, Surya Sumpeno, and Yoyon Kusnendar Suprapto. Community feedback analysis using latent semantic analysis (lsa) to support smart government. In *2019 International Seminar on Intelligent Technology and Its Applications (ISITIA)*, pages 428–433. IEEE, 2019.

[51] A. Soufan. Deep learning for sentiment analysis of arabic text. 2019.

[52] Nandhini Kumaresh Venkateswarlu Bonta and N. Janardhan. A comprehensive study on lexicon based approaches for sentiment analysis. 2019.

[53] Y. Q. Lu X. Li and D. Tao. "robust subspace clustering by cauchy loss function," ieee transactions on neural networks and learning systems. *vol. 30, no. 7, pp. 2067–2078*, 2019.

[54] Bei Yu. An evaluation of text classification methods for literary study. *Literary and Linguistic Computing*, 23(3):327–343, 2008.